UNIVERSIDAD CARLOS III DE MADRID

Escuela Politécnica Superior

Ingeniería Industrial

Departamento de Automática

Reconstrucción de escenas 3D a partir de un conjunto de vistas tomadas desde una cámara

Autor: Javier Rosado Sánchez-Izquierdo

Tutor: Arturo de la Escalera Hueso

Junio de 2012

Reconstrucción de escenas 3D a partir de un conjunto de vistas tomadas desde una cámara

Javier Rosado Sánchez-Izquierdo

Junio de 2012

A mi familia, por su apoyo y sus ánimos a lo largo de toda mi vida.

Resumen

La visualización de imágenes en 3D, o el efecto recreado de este tipo de visión, es posible gracias a los sistemas estereoscópicos, o la simulación del efecto que éstos producen, que nos permiten obtener diferentes vistas de una misma escena, y que, mediante el procesado de dicha información, se puede extraer información de la profundidad de los elementos que componen la escena, pudiendo así, realizar este efecto.

En este proyecto, se ha realizado la implementación de un sistema de visión, que a partir de la captura de una serie de imágenes de una escena, con una cámara convencional, sea capaz de simular un efecto estereoscópico, y así poder realizar una reconstrucción tridimensional de la escena.

El software capaz de realizar esta tarea, ha sido implementado en C/C++, y utiliza como apoyo, las librerías de visión por computador OpenCV.

Para obtener el efecto 3D, este proyecto se ha basado en la teoría de los algoritmos del Structure From Motion, que se basan en la reconstrucción de escena 3D, a partir del movimiento de la escena (ya sea del entorno, o de la cámara).

El sistema se compone de dos módulos principales. El primero de ellos, permite realizar la calibración de la cámara, y el segundo, se encarga de realizar la rectificación, búsqueda de correspondencia y procesado de los pares de imágenes, y de realizar la reconstrucción 3D de la escena.

Agradecimientos

En primer lugar quisiera agradecerle a mi tutor Arturo la posibilidad de realizar este proyecto, así como su dedicación y ayuda cuando la he necesitado.

A continuación quiero agradecer a mis padres y a mi hermano todo el apoyo que me han dado en todas las decisiones que he tomado en la vida, que aunque no siempre han sido las más acertadas, siempre han estado ahí para ayudarme cuando lo he necesitado, y sin los cuales, todo lo que soy hoy y lo que tengo, nunca hubiese sido posible.

Gracias a Andrea por todo el cariño y apoyo que me ha dado estos años, su espíritu y su ánimo me han dado las fuerzas que muchas veces he necesitado para seguir adelante cuando me he encontrado con una dificultad, su alegría que me transmitía en los momentos difíciles y su apoyo en las decisiones que he tomado. Nunca podré agradecerle lo suficiente todo lo que me ha aportado.

A mis compañeros y amigos Almudena, Ana, Bea, Bruno, Gloria, Laura, Laura, Manu, Mayte, Moisés y Rubén, que aunque algunos estuvieron conmigo desde el principio y otros fueron apareciendo en el camino, sin ellos nunca hubiese podido pasar por esta etapa de mi vida. Porque tanto en los muchos buenos ratos que hemos pasado juntos, como en los pocos malos, han sido un gran apoyo y siempre han sido capaces de sacarme una sonrisa cuando no todo iba como nos gustaría. Nunca olvidaré todo lo que hemos compartido y hemos pasado juntos.

A mis amigos Adri, Alberto, Blanca, Chema, Edu, Guille, Jara, Lore, Miguel, Mundi, Rut y Santi por todos los años que hemos compartido buenos momentos, y aunque no todos están ahora, siempre guardaré los buenos momentos que hemos vivido juntos y los que nos quedan por vivir.

No puedo terminar de agradecer, sin mencionar a Salva, Buira, Edu, Yago, Oso, Conde, Nacho, Edu, Cristian, Antonio, Juan, Vero, Laura, Irene, Sofía y Marina que han estado ahí cuando más lo he necesitado, y me han hecho olvidarme de los pequeños problemas y me han hecho pasar tantos buenos ratos que espero que nunca terminen.

Al resto de miembros de mi familia que no nombro, pero que tengo muy presentes, porque por ellos también ha sido posible llegar hasta aquí. Así como a mi cuñada Bea por los buenos ratos que hemos pasado juntos, y los que nos quedan por pasar.

También quiero dar las gracias a los que ya no están, pero que tengo presente siempre en mi cabeza, porque a ellos también va dedicado este proyecto.

INDICE

Índice de figuras

Índice de tablas

Capítulo 1: Introducción

Hace años el término visión por computador podría sonar a ciencia ficción, pero en las últimas décadas, la visión de los ordenadores, es una realidad.

El principal objetivo de la visión por computador es tomar decisiones útiles y eficaces sobre objetos reales a partir de la información que percibe el computador. Es, por tanto, necesario construir una descripción del modelo para cada imagen. La visión artificial tiene como finalidad la extracción de información del mundo físico a partir de imágenes, utilizando para ello un computador. Se trata de un objetivo ambicioso y complejo que actualmente se encuentra en una etapa de desarrollo, aun lejos del estado del arte.

Este mundo es percibido a través de cámaras digitales las cuales modelan el mundo real utilizando los principios de la geometría. Una característica de los principios fundamentales de los sistemas presentes y futuros se basa en la comprensión espacial del mundo que percibe el computador. El dominio de la relación espacial, medición del espacio tridimensional, la propagación de la luz a través de una lente y el modelo matemático de formas y tamaños de objetos son los verdaderos fundamentos de esta disciplina.

Uno de los aspectos más sorprendentes cuando se estudia la percepción humana es la capacidad del observador para determinar la estructura tridimensional de los objetos a partir de patrones bidimensionales de luz. Los primeros estudios enfocados al análisis tridimensional a partir de imágenes fueron realizados por la comunidad científica de la fotogrametría. Estos estudios fueron retomados por científicos de la visión por computadora hasta llegar a desarrollar enfoques diferentes para una misma problemática. Hoy en día existe un interés por parte de los fotográmetras por comprender la capacidad de dichos algoritmos para poder derivar mediciones tridimensionales de alta precisión. Cuando se intenta evaluar una aproximación computacional para la percepción artificial de formas tridimensionales es necesario tener en cuenta dos hechos.

Por una parte que existen numerosos atributos de la estructura tridimensional que potencialmente podrían estar representados en el sistema visual (curvatura, distancia relativa, orientación local, entre otras) cuyas dificultades computacionales no son las mismas, y por otra, que para la evaluación de las diferentes aproximaciones computacionales es necesario examinar la validez de las hipótesis subyacentes. Puesto que existen numerosas escenas que producen la misma imagen, todos los análisis

computacionales de la percepción tridimensional deben restringirse a un conjunto de posibles interpretaciones suponiendo una serie de restricciones más o menos reales.

En cualquier caso, el recorrido que se propone a continuación es a través de los modelos que han sido propuestos para analizar la estructura de objetos tridimensionales a partir de diferente información. Estudiaremos la estimación de la forma a partir de la realización de un efecto de visión estéreo. Si tenemos en cuenta que el tipo de sensor que normalmente proporciona a un sistema de visión artificial la información del mundo tridimensional que le rodea es una cámara de fotos o de video, nuestro primer objetivo en visión 3D debe ser explicar cómo usar la información bidimensional que recogen nuestros sensores (cámara digital) que tenemos sobre la escena, para realizar medidas del mundo 3D subyacente. El modelo geométrico de proyección 3D a 2D más utilizado es el modelo de proyección Pin-Hole.

Los algoritmos que reconstruyen la estructura 3D de una escena o calculan la posición de objetos en el espacio necesitan las ecuaciones que unen los puntos tridimensionales con sus correspondientes proyecciones bidimensionales, y aunque dichas ecuaciones vienen proporcionadas por la ecuación de proyección, es normal suponer que los puntos del mundo real (tridimensionales) pueden venir dados con relación a un sistema de coordenadas distintas del proporcionado por el sistema de referencia de la Cámara, siendo necesario además relacionar las coordenadas de un punto en la imagen con las coordenadas correspondientes en el sistema de referencia proporcionado por la cámara.

La tecnología dedicada a la visualización de imágenes y video está en constante desarrollo. Una parte que está evolucionando notablemente en los últimos años, es la introducción del 3D en todos los sistemas audiovisuales que hay en el mercado. El 3D es cada vez más demandado y últimamente se intenta introducir en dispositivos domésticos.

Lo que nos lleva a plantearnos el objeto de este proyecto.

1.1 Objetivos

Este proyecto en el que nos embarcamos, tiene como principal objetivo el diseño e implementación de un programa informático (programado en lenguaje C++) capaz de realizar una reconstrucción tridimensional de una escena, a partir de un conjunto de imágenes bidimensionales, tomadas con una cámara de fotos convencional (cámara digital). Para ello nos vamos a basar en los algoritmos del SFM (Structure From Motion) y como soporte, nos vamos a ayudar de las librerías de visión por computador OpenCV.

1.2 Estructura del proyecto

El proyecto se ha dividido en diversas secciones que facilitan la comprensión del mismo para el lector, así como una forma más sencilla de saltarse las partes que puedan o no interesar en un momento determinado, siendo independientes entre ellas.

En el primer capítulo, como se ha podido comprobar hasta el momento, se hace una breve introducción al mundo de la visión por computador, así como los objetivos de este proyecto.

En el segundo capítulo, lo que nos encontramos es una explicación de todos los aspectos teóricos, así como las ecuaciones matemáticas y físicas, en las que se basa el proyecto. En este capítulo, se encuentra completamente separada cada parte que compone la estructura del proyecto, desde la calibración de la cámara, hasta la reconstrucción final de la escena.

En el tercer capítulo, se hace una explicación detallada del proceso seguido para la realización del proyecto, pudiendo comprobarse la teoría en la que se basa, así como las funciones más importantes que se han ido utilizando para el desarrollo de la solución software.

En el cuarto capítulo se detallan algunos ejemplos de la aplicación, así como los resultados que se han ido obteniendo de cada módulo y cada estudio realizado.

En el último capítulo se muestran las conclusiones del trabajo realizado, así como futuras líneas de investigación y trabajo, y algunas ideas susceptibles de ser realizadas en otros proyectos.

Capítulo 2: Antecedentes/Estado de la cuestión

2.1 Structure From Motion (SFM)

Los algoritmos SFM son capaces de calcular la estructura 3D de una escena a partir de varias proyecciones de la misma, es decir, a partir de una secuencia de vídeo. Esta reconstrucción se puede usar para calcular la pose de la cámara a lo largo de la secuencia. Por lo general no funcionan en tiempo real, por lo que no pueden usarse en aplicaciones de realidad aumentada, pero sí en aplicaciones de vídeo aumentado.

En este capítulo se va a proceder a describir técnicas y herramientas para obtener información sobre la geometría tridimensional a partir de imágenes bidimensionales. Esta tarea no es trivial ya que el proceso de formación de las imágenes no es generalmente invertible.

Una posibilidad es aprovechar el conocimiento sobre la escena, con el fin de reducir el número de grados de libertad. Por ejemplo, las restricciones que imponen el paralelismo y la coplanaridad, pueden utilizarse para reconstruir formas de geometrías simples como líneas y polígonos planos, a partir de sus posiciones proyectadas en vistas simples.

Otra posibilidad es el uso de la correspondencia entre puntos de la imagen que aparezcan en múltiples imágenes.

Dadas estas dos o más vistas, un punto tridimensional, puede ser reconstruido mediante triangulación. Un prerrequisito importante es la determinación de la calibración de la cámara, la determinación de la pose, que puede ser representada por una matriz de proyección. La teoría geométrica del SFM permite calcular las matrices de proyección y los puntos 3D, utilizando solo la correspondencia entre los puntos de cada imagen.

Las técnicas del SFM se utilizan en una gran cantidad de aplicaciones, incluyendo reconstrucción fotogramétrica, la reconstrucción automática de modelos de realidad virtual a partir de secuencias de video, la reconstrucción arquitectónica, secuencias de realidad aumentada y video aumentado.

Para llevar a cabo este proceso, se tienen que seguir una serie de pasos.

Una primera cuestión, es la calibración de la cámara, la cual nos permite obtener los parámetros internos de la cámara. Una vez calibrada la cámara, el siguiente paso lógico es el de obtener la rectificación de las imágenes tomadas por la cámara, con el fin de eliminar distorsiones y otras aberraciones producidas por la imperfección del sensor y la lente de la cámara.

Una vez disponemos de las imágenes correctamente rectificadas, podemos proceder a realizar la correspondencia entre los puntos de dos imágenes. Esto nos permite calcular su geometría epipolar. Esta relación se representa algebraicamente por la matriz fundamental, la cual puede descomponerse para recuperar una pareja de matrices de proyección.

Cuando ya disponemos de las matrices de proyección, la relación existente entre las coordenadas 3D de los puntos puede obtenerse mediante triangulación.

Un último apartado en los algoritmos del SFM es lo que se conoce como Bundle Adjustment, que se utiliza para obtener unos parámetros de máxima verosimilitud de los valores, mediante métodos de optimización no lineales.

2.2 Calibración de la cámara

2.2.1 Introducción

En aplicaciones con visión por computador un aspecto importante a considerar es el proceso de calibración de la cámara, el cual consiste en determinar los parámetros internos de la cámara, tales como distancia focal, factores de distorsión y puntos centrales del plano imagen.

Existen diferentes métodos de realizar este proceso, los cuales difieren en la forma de capturar desde las imágenes los parámetros intrínsecos.

El proceso de calibración de cámaras, es necesario para poder extraer información métrica a partir de imágenes 2D del mundo 3D. El proceso de calibración de cámara abre la posibilidad de realizar aplicaciones efectivas en visión, tal como realidad aumentada, reconocimiento, seguimiento y reconstrucción 3D, los cuales se basan en el conocimiento de calibración y pose de la cámara.

Estos parámetros normalmente son calculados desde un patrón de calibración que contiene rasgos fácilmente detectables de manera precisa en la imagen capturada.

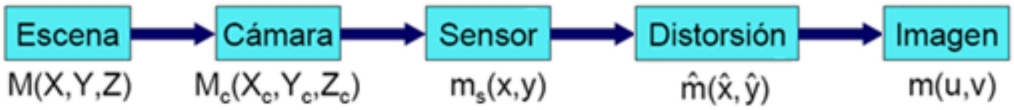


Figura 2.2.1: Diagrama de bloques del proceso de calibración

2.2.2 Fundamento teórico

En el proceso de calibración, se considera que el paso desde las coordenadas de un punto en el sistema del mundo hasta las coordenadas con que aparece en el sistema de la cámara puede descomponerse en las siguientes etapas:

- Paso de escena (3D) a cámara (óptica)
- Paso de cámara (óptica) a sensor
- Distorsión de la lente
- Paso de sensor a imagen (2D)

2.2.2.1. El modelo de cámara.

Existen varios modelos que definen la geometría que indica cómo un objeto del mundo real se proyecta sobre un plano (Ej.: sensor CCD), convirtiéndose en una imagen 2D. El más sencillo se basa en el concepto de cámara oscura (Pin-Hole).

Tal y como se puede apreciar en la Figura 2.2.2, una cámara oscura es una caja que permite a los rayos de luz penetrar en su interior a través de un pequeño agujero e incidir sobre una superficie fotosensible. Por tanto, un objeto situado a una distancia Z respecto a la pared agujereada, se ve proyectado de forma invertida en el plano imagen, situado a una distancia f de dicha abertura.

El tamaño de este agujero tan solo permite el paso de una muy pequeña cantidad de luz, es por ello que el uso de lentes es necesario en la práctica. Esto permite un incremento del número de rayos que la atraviesan, de manera que se facilita la generación de la imagen en condiciones lumínicas adversas. En este caso varía ligeramente la forma en la que los rayos llegan al plano imagen, pero el modelo de cámara oscura sigue considerándose como válido, aún cuando se usen una o más lentes.

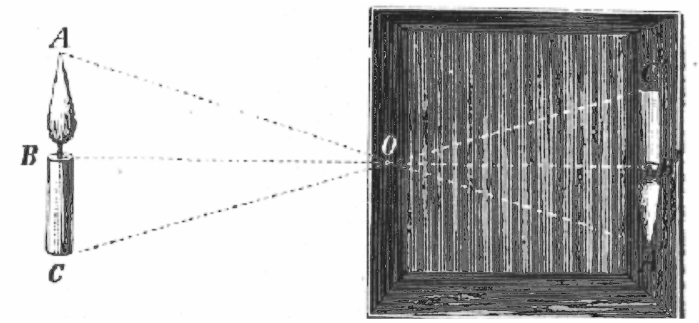


Figura 2.2.2: Principio óptico de cámara oscura

En la Figura 2.2.3 se puede ver una representación geométrica equivalente del modelo de cámara oscura, la cual facilitará la matemática empleada a partir de ahora.

Estas son algunas consideraciones a tener en cuenta:

- El centro óptico *C* es el punto de la lente tal que cualquier rayo de luz que pasa por él no sufre desviación. Por el momento se considerará como origen de coordenadas del sistema.

- El eje óptico es la línea imaginaria que parte del centro óptico y corta perpendicularmente al plano imagen.

- La distancia focal *f* es la distancia existente desde el centro óptico al plano focal.

- El plano focal o plano imagen se sitúa en $Z = f$. Es el plano virtual donde se forma la imagen sin inversión alguna.

- El punto principal *p*, es la intersección del eje óptico con el plano imagen. Lo habitual, es que no esté alineado con el centro del sensor de la cámara. Este hecho se comentará detalladamente en la sección 2.2.2.2.

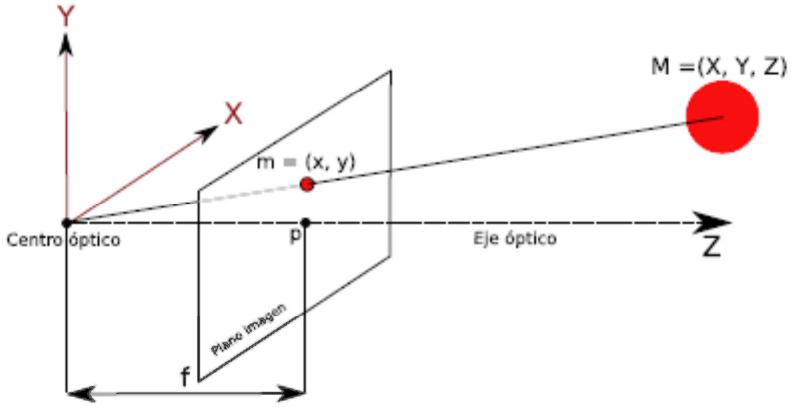


Figura 2.2.3: Representación geométrica del modelo matemático basado en la cámara oscura.

- Si se observa la geometría anterior desde la dirección del eje X resulta la Figura 2.2.4(a). Análogamente si se observa desde la dirección del eje Y, resulta la Figura 2.2.4(b). Debido a que en ambos casos se cumple una relación de semejanza de triángulos, se puede afirmar lo siguiente.

$$\frac{y}{f} = \frac{Y}{Z} \qquad \frac{x}{f} = \frac{X}{Z} \tag{2.1}$$

$$(X, Y, Z) \Rightarrow \left(\frac{fX}{Z}, \frac{fY}{Z}\right) \tag{2.2}$$

Según este modelo, todo punto *M* del mundo real, se transformará en un punto *m* de una imagen según la relación

$$m \approx PM \tag{2.3}$$

donde *P*, es la matriz de proyección, que será descrita en detalle a lo largo del apartado siguiente.

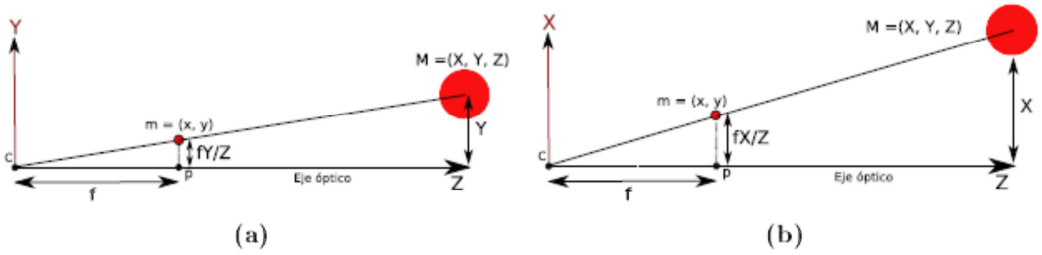


Figura 2.2.4: Modelo de cámara oscura visto desde la dirección del eje *X* y eje *Y* respectivamente.

2.2.2.2. Parámetros intrínsecos.

La matriz de proyección ***P*** de la ecuación (2.3), permite transformar las coordenadas de un punto 3D del mundo real en píxeles de una imagen. Se construye a partir de una matriz ***K*** y un vector de valores nulos,

$$P = [K|0] \tag{2.4}$$

donde la *K* es la matriz de calibración, la cual está formada por una serie de parámetros, denominados parámetros intrínsecos:

$$K = \begin{pmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \tag{2.5}$$

- c_x y c_y indican el desplazamiento del centro de coordenadas del plano imagen, respecto al punto principal. Será nulo sólo si el eje óptico coincide con el centro del sensor de la cámara, pero como ya se comentó en la sección 2.2.2.1, el eje óptico no siempre atraviesa el centro de la imagen generada.
- El factor *s* (skew factor) determina el grado de perpendicularidad de las paredes de los píxeles del sensor. Es inversamente proporcional a la tangente del ángulo que forman los ejes *X* e *Y* , por lo que *s* tendrá un valor nulo si los píxeles son rectangulares. Esto suele ser así en casi todos los sensores utilizados hoy en día.

- f_x y f_y son dos distancias focales en píxeles. Son proporcionales a la longitud focal *f* considerada en las ecuaciones (2.1) y (2.2), según:

$$f_x = f S_x \tag{2.6}$$

$$f_y = f S_y \tag{2.7}$$

donde:

- *f* es la longitud focal física de la lente, en unidades de longitud (milímetros, micras, etc.).

- S_X y S_Y son el número de píxeles por unidad de longitud del sensor, a largo del eje x y del eje y respectivamente. Como es obvio, si el sensor tiene el mismo número de píxeles por unidad de longitud en todas sus dimensiones, las dos focales f_X y f_Y tendrán el mismo valor.

2.2.2.3. Distorsiones.

Para que la relación (2.3) sea válida en la práctica, es necesario tener en cuenta las distorsiones que se producen durante la formación de las imágenes.

Distorsión radial

El uso de lentes facilita la entrada de luz, un enfoque adecuado y una mayor versatilidad, pero también introduce deformaciones en las imágenes que se forman en el sensor. Uno de estos efectos es la distorsión radial o distorsión en barrilete, y se muestra esquemáticamente en la Figura 2.2.5. Esta distorsión es debida a que algunas lentes provocan que los rayos más alejados del centro óptico, se curven mucho más que aquellos que inciden directamente en las proximidades del centro de la lente.

Este tipo de distorsión se hace más acusada en la zona próxima a los límites de las imágenes, tal y como puede apreciarse en el ejemplo de la Figura 2.2.6. Es importante remarcar que también crece según disminuye la longitud focal de la lente usada o cuando se usan ópticas de mala calidad.

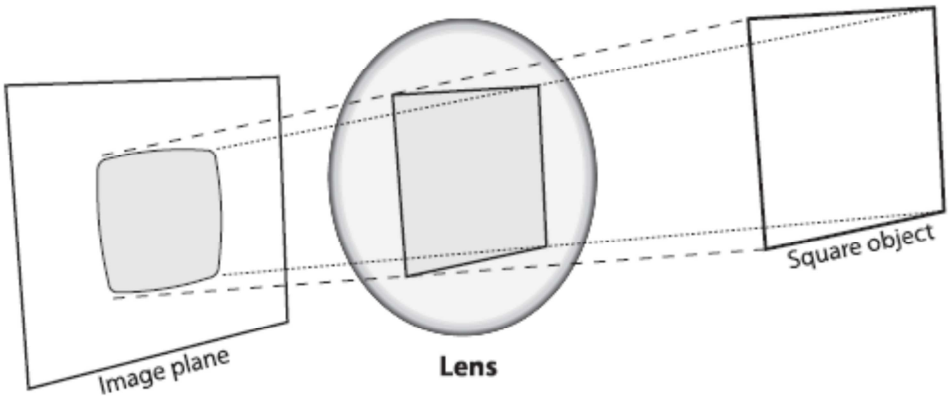


Figura 2.2.5: Esquema de la distorsión radial producida en una cámara

La distorsión radial puede ser modelada mediante la serie de Taylor en torno a $r = 0$ mostrada en [6]. De esta forma las coordenadas de una imagen con distorsión quedan como sigue,

- $(x_{dist}; y_{dist})$ es la posición de un píxel en la imagen distorsionada.
- $(\tilde{x}, \tilde{y})$) es la posición de un píxel en la imagen corregida.

- $\tilde{r}$ es la distancia desde el centro que se expresa como $\sqrt{\tilde{x}^2 + \tilde{y}^2}$.
- $L(\tilde{r})$ es el factor de distorsión y es igual a

$$L(\tilde{r}) = (1 + \mathrm{k}_1\mathrm{r}^2 + \mathrm{k}_2\mathrm{r}^4 + \mathrm{k}_3\mathrm{r}^6 + \ldots) \qquad (2.8)$$

donde los k_i son los coeficientes de distorsión.

En la práctica sólo suele ser necesario el uso de los coeficientes k_1 y k_2. Por el contrario, para aquellas lentes que introduzcan mucha distorsión (Ej.: lentes baratas, ojos de pez, etc), será necesario tener en cuenta k_3 para que la corrección se haga adecuadamente.

Distorsión tangencial

Otro tipo de distorsión que se ha de tener en consideración a la hora de trabajar con imágenes es la distorsión tangencial. En este caso las deformaciones producidas se deben a un problema propio de la cámara, no de la lente.

Figura 2.2.6 Distorsión radial corregida

Puede ocurrir que durante el proceso de fabricación, el sensor no quede perfectamente pegado a la pared sobre la que se apoya. Esta situación se presenta en la Figura 2.2.7. De esta forma la lente usada no estará paralela al plano donde se forma la imagen.

La distorsión tangencial se puede describir de forma sencilla con dos parámetros $p1$ y $p2$, los cuales transforman las coordenadas de la imagen según dictan las siguientes ecuaciones [9]

$$\tilde{x} = x_{dist} + (2p_1 y_{dist} + p_2(r^2 + 2x_{dist}^2) \quad (2.9)$$

$$\tilde{y} = y_{dist} + (2p_1 x_{dist} + p_2(r^2 + 2y_{dist}^2) \quad (2.10)$$

2.2.2.4. Parámetros extrínsecos.

En el apartado 2.2.2.1 se asumió el hecho de que el centro óptico ***C*** era el origen de coordenadas del mundo. Esto era así a efectos del cálculo de parámetros intrínsecos, con lo cual el modelo era válido siempre que el sistema no fuera movido de su posición inicial.

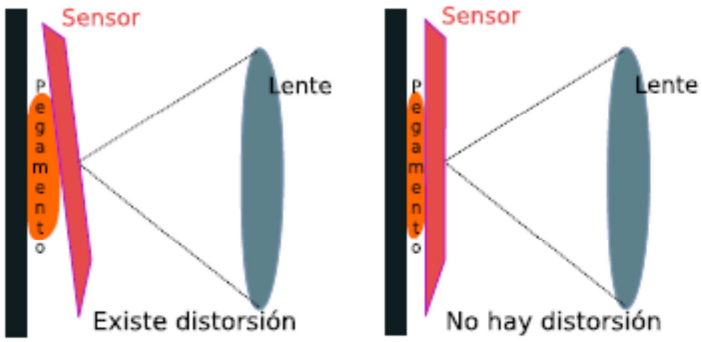


Figura 2.2.7: Efecto que provoca la distorsión tangencial en la cámara.

Por otro lado, en la mayor parte de las aplicaciones prácticas es necesario que la cámara se mueva o se gire, para captar adecuadamente la escena.

Por ello, para poder modelar el sistema con independencia de que su posición haya sido alterada o de que un objeto se pueda referenciar respecto al origen de coordenadas de la cámara, es necesario modificar la ecuación (2.3) introduciendo una nueva matriz ***W***:

$$m \approx PWM = P'M \qquad (2.11)$$

$$W = [R \;\; t] \qquad (2.12)$$

donde:

- ***R*** es una matriz de rotación, que representa un giro de la cámara (o de un objeto respecto de ella). Tendrá una forma distinta dependiendo de respecto a que eje (X, Y, Z) se haga la rotación:

$$R_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\Psi & sen\Psi \\ 0 & -sen\Psi & cos\Psi \end{pmatrix} \qquad (2.13)$$

$$R_y = \begin{pmatrix} \cos\varphi & 0 & -\text{sen}\,\varphi \\ 0 & 1 & 0 \\ \text{sen}\,\varphi & 0 & \cos\varphi \end{pmatrix} \qquad (2.14)$$

$$R_z = \begin{pmatrix} \cos\theta & -\,\text{sen}\,\theta & 0 \\ \text{sen}\,\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad (2.15)$$

Si el giro se realiza respecto al eje *Z*, tal y como se dibuja en la Figura 2.2.8, las nuevas coordenadas quedarán de la siguiente forma:

$$\begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \Leftrightarrow \begin{cases} X' = X\cos\theta + Y\sin\theta \\ Y' = -X\sin\theta + Y\cos\theta \\ Z' = Z \end{cases} \qquad (2.16)$$

Si el giro se hiciera respecto a los otros dos ejes (*X* o *Y*), la operación se haría de forma análoga, utilizando para ello la pertinente matriz de rotación *R*. Además se observaría que la componente correspondiente a dicho eje de rotación tampoco se vería alterada.

- **t** es un vector de translación que representa un desplazamiento del sistema de coordenadas. Permite indicar un cambio de posición de la cámara o de un objeto respecto a ella. De esta forma $\mathbf{M}_{\mathbf{final}} = \mathbf{M}_{\mathbf{inicial}} - \mathbf{t}$.

1. Una rotación en tres dimensiones se define con la variación de tres ángulos (Ψ,φ,θ).

2. Una translación en el espacio se especifica con tres parámetros (x, y, z).

3. Los parámetros intrínsecos propios de la cámara son cinco (f_x, f_y, c_x, c_y, s).

Se concluye que es necesario conocer estos once parámetros de cada imagen que haya generado una misma cámara, si se utiliza la expresión (2.11). Lo que se verá en el siguiente apartado, es el método que se ha seguido en este proyecto para la obtención de dichos parámetros.

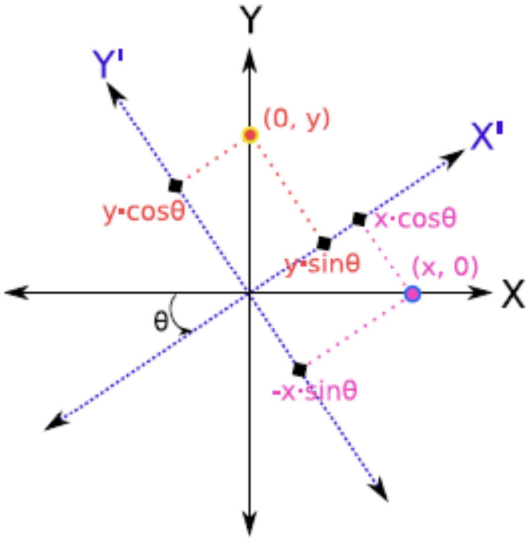


Figura 2.2.8: Rotación de la cámara respecto al eje Z.

2.2.2.5 Número de imágenes necesarias para la calibración

En la parte final de este capítulo, se deducirán cuantas capturas del damero son necesarias para obtener unos parámetros de calibración precisos. Así mismo se establecerán las dimensiones adecuadas del objeto de calibración.

Considerando que el damero tiene *N* esquinas interiores y que se tienen *J* imágenes:

- Las *J* imágenes del tablero de ajedrez imponen un total de 2*NJ* restricciones, una por cada esquina detectada.

- 5 son los parámetros intrínsecos y 6*J* los parámetros extrínsecos a calcular durante el proceso.

Si se resuelve la ecuación $2NJ \geq 6J + 5$, se obtiene que para $J = 2$ tan solo se necesitarían $N = 5$ esquinas.

En la práctica, se ha comprobado que es necesario un número superior a 2 imágenes para obtener una calibración precisa.

2.2.3 Método de calibración

Existen diferentes métodos de calibración de cámaras, entre los más conocidos se encuentran el de Tsai [12], Faugeras [13] y Zhang [10]. El método de Tsai se basa en el método de Pin-Hole, anteriormente explicado, y corrige la distorsión a partir de un solo coeficiente, que únicamente corrige la distorsión radial. El sistema general del método de Tsai plantea 9 incógnitas, de las cuales 6 son de parámetros extrínsecos y 3 de intrínsecos.

El método de Faugeras propone el cálculo de los parámetros intrínsecos y extrínsecos a partir de un conjunto de puntos de control. Conocidas las coordenadas 3D de dichos puntos y las coordenadas 2D de la imagen se podrán obtener los parámetros de calibración. El proceso de calibración se realiza en dos pasos, primero se obtiene la matriz de proyección y después se obtienen los valores de parámetros intrínsecos y extrínsecos a partir de la matriz de proyección obtenida anteriormente. Este método no modela la distorsión.

El método de Zhang propone una técnica de calibración que se basa en la observación de un patrón de referencia del que se toman varias imágenes desde diferentes posiciones, para no tener que preparar y medir los puntos de la escena. La ventaja de este método, es que se pueden obtener los parámetros de las cámaras sin necesidad de conocer la posición de los puntos de la escena. El proceso de calibración se realiza en tres pasos, primero se transforma el sistema de coordenadas del mundo en el de la cámara (matriz extrínsecos), después se realiza una corrección de la distorsión y luego se obtienen las coordenadas 2D de la imagen (matriz intrínsecos).

El método de Faugeras tiene la desventaja de que no tiene en cuenta los valores de distorsión introducidas por las lentes y la matriz de parámetros intrínsecos obtenidos son con distorsión. En oposición, el método de Zhang hace una corrección de esta distorsión y los valores de la matriz de intrínsecos son los obtenidos después de haber realizado la corrección de dicha distorsión.

En este proyecto se ha optado por seguir las directrices mostradas en [3], que se basan en el algoritmo de calibración de Zhengyou Zhang [10] y en sus posteriores mejoras [11]. La elección de este método se debe a varios motivos:

- Facilidad de implementación: el uso de librerías de visión OpenCV [3] ha permitido su programación sin mucha dificultad.

- Sencillez de ejecución: tan sólo es necesario realizar varias capturas de un objeto conocido, de manera que éste varíe su posición a lo largo de una serie de fotografías.

- Validez de los resultados: los datos arrojados tras la calibración son tan válidos como los que se obtienen con otros métodos [10], sin que se requiera un coste computacional elevado o inversión en caros equipos.

2.2.3.1. Procedimiento

Como paso previo a la ejecución que permite obtener la calibración, ha sido necesario el uso de un tablero de ajedrez que se ha imprimido sobre una superficie plana. Este objeto de calibración sigue un patrón sencillo de zonas blancas y negras claramente diferenciadas, lo cual permite al algoritmo trabajar eficazmente e identificar regiones de forma precisa.

A continuación se describirán los pasos que se han llevado a cabo para el cálculo de los parámetros intrínsecos y extrínsecos:

1. Realización de una serie de capturas como las de la Figura 2.2.9. Se ha de procurar que el objeto de calibración varíe su posición y recorra la mayor parte del encuadre.

De este modo, un número considerable de puntos se situará en las zonas más sensibles a la distorsión, y así se logrará una mayor precisión a la hora de calcular los parámetros de distorsión y corregir la imagen.

2. Detección automática de la posición de las esquinas interiores del damero. Para ello el algoritmo necesitará:

- Las imágenes donde se muestra el tablero que convertirá a blanco y negro antes de procesarlas.
- El número de esquinas interiores que contiene el objeto de calibración. Se le indicará cuántas hay en cada lado de forma independiente.

Si tras el proceso de búsqueda se detectaran todas las esquinas interiores de forma satisfactoria, se obtendrían resultados similares a los mostrados en la Figura 2.2.10.

3. Cálculo de los parámetros intrínsecos y extrínsecos mediante la resolución de una homografía entre puntos.

4. Cálculo de los parámetros de distorsión que permitirán corregir posibles aberraciones de la imagen.

Los dos últimos pasos se explicarán detalladamente a lo largo de los siguientes dos apartados.

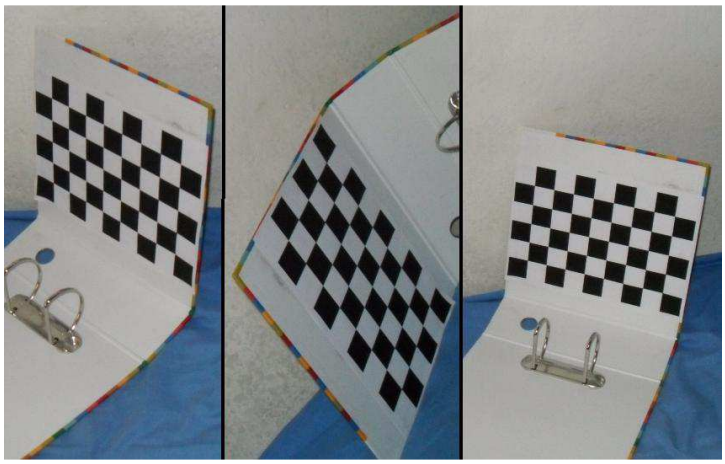


Figura 2.2.9: Secuencia parcial de imágenes para la calibración.

2.2.3.2 Cálculo de los parámetros extrínsecos e intrínsecos.

Para el cálculo de ambas familias de parámetros se asume primeramente una relación de homografías entre los puntos detectados del tablero y los puntos de la imagen formada en el sensor de la cámara, según se muestra en la Figura 2.2.11. Una homografía de este tipo tiene una expresión tal que:

$$\boldsymbol{m} = \alpha H \boldsymbol{M} \qquad (2.17)$$

Si se compara con la ecuación (2.11), se podrá apreciar que $\alpha H \Leftrightarrow PW$, donde *alpha* no es más que un mero factor de escala.

$$\boldsymbol{m} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \alpha KWM = \alpha \begin{pmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} (r_1 \;\; r_2 \;\; r_3 \;\; t) \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \qquad (2.18)$$

Figura 2.2.10: Imágenes detectadas correctamente por el algoritmo de calibración.

donde $\mathbf{r_1}$, $\mathbf{r_2}$ y $\mathbf{r_3}$ son las columnas de la matriz de rotación *R*, **t** el vector de translación y el resto de parámetros los ya comentados con anterioridad en la sección 2.2.2.2 Como los puntos 3D del objeto de calibración se encuentran en el mismo plano, es posible simplificar la ecuación anterior estableciendo que la coordenada *Z* sea nula. Esto provoca que el término $\mathbf{r_3}$ no afecte en modo alguno a la expresión, la cual queda reducida a

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \alpha K \begin{pmatrix} r_1 & r_2 & t \end{pmatrix} \begin{pmatrix} X \\ Y \\ 0 \\ 1 \end{pmatrix} \qquad (2.19)$$

Por todo lo anterior, se concluye que la homografía *H* existente entre los puntos del damero y los detectados en la imagen generada, puede expresarse como

$$H = \begin{pmatrix} h_1 & h_2 & h_3 \end{pmatrix} = \alpha K \begin{pmatrix} r_1 & r_2 & t \end{pmatrix} \qquad (2.20)$$

deduciéndose de la ecuación anterior el siguiente sistema

$$h1 = \alpha K r_1 \qquad h2 = \alpha K r_2 \qquad h3 = \alpha K t \qquad (2.21)$$

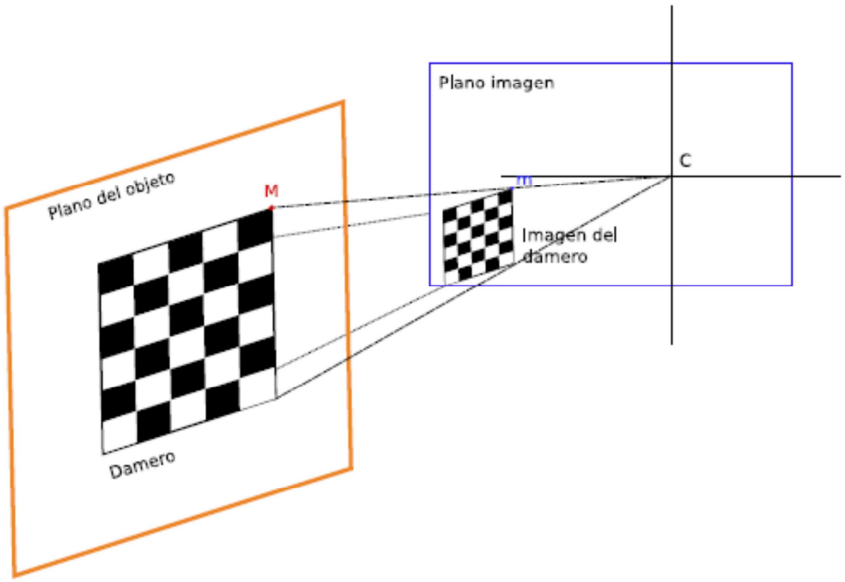


Figura 2.2.11: Homografía existente entre cualquier punto imagen y punto objeto.

Como toda matriz de rotación *R* es ortonormal, el módulo de cualquiera de sus columnas **ri** valdrá uno y además serán ortogonales entre sí. Utilizando estas propiedades y lo anteriormente expuesto [9], se derivan el siguiente par de restricciones

$$h_1^T K^{-T} K^{-1} h_2 = 0, \qquad (2.22)$$

$$h_1^T K^{-T} K^{-1} h_1 = h_2^T K^{-T} K^{-1} h_2, \qquad (2.23)$$

Por simplificar la expresión anterior definimos:

$$B = K^{-T} K^{-1} = \begin{pmatrix} B_{11} & B_{12} & B_{13} \\ B_{12} & B_{22} & B_{23} \\ B_{13} & B_{23} & B_{33} \end{pmatrix} \qquad (2.24)$$

Finalmente se muestran las expresiones [9] que permitirán al algoritmo de calibración calcular los parámetros intrínsecos

$$f_x = \sqrt{\frac{\lambda}{B_{11}}} \tag{2.25}$$

$$f_y = \sqrt{\frac{\lambda B_{11}}{B_{11}B_{22} - B_{12}^2}} \tag{2.26}$$

$$c_x = \frac{sc_y}{f} - B_{13}\frac{f^2}{\lambda} \tag{2.27}$$

$$c_y = \frac{B_{12}B_{13} - B_{11}B_{23}}{B_{11}B_{22} - B_{12}^2} \tag{2.28}$$

$$\lambda = \alpha^{-1} = B_{33} - \frac{B_{13}^2 + c_y(B_{12}B_{13} - B_{11}B_{23})}{B_{11}} \tag{2.29}$$

$$s = -\frac{B_{12}f_x^2 f_y}{\lambda} \tag{2.30}$$

Por otra parte, retomando las ecuaciones que definen la homografía y siendo conocidos los parámetros intrínsecos, se pueden obtener fácilmente los parámetros extrínsecos:

$$r_1 = \lambda K^{-1}h_1 \tag{2.31}$$

$$r_2 = \lambda K^{-1}h_2 \tag{2.32}$$

$$r_3 = r_1 \times r_2 \tag{2.33}$$

$$t = \lambda K^{-1}h_3 \tag{2.34}$$

2.2.3.3 Cálculo de los parámetros de distorsión.

Una vez obtenidos los parámetros intrínsecos, y habiendo supuesto que no existían distorsiones:

$$\begin{pmatrix} x_{sindist} \\ y_{sindist} \end{pmatrix} = (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \begin{pmatrix} x_{dist} \\ y_{dist} \end{pmatrix} + \begin{pmatrix} 2p_1 x_{dist} y_{dist} + p_2(r^2 + 2x_{sindist}^2) \\ 2p_2 x_{dist} y_{dist} + p_1(r^2 + 2y_{sindist}^2) \end{pmatrix} \qquad (2.35)$$

Aplicando un algoritmo de minimización no lineal se podrán obtener los parámetros de distorsión. Inicializándolos a cero, se procederá al cálculo de nuevos valores de forma iterativa. El algoritmo analizará cuando los parámetros calculados permitan obtener una imagen en la que el error debido a la distorsión sea próximo a cero.

2.3 Geometría Epipolar

Es posible determinar sin incertidumbre, con una sola cámara, cómo un punto 3D del mundo real se transforma en un punto 2D perteneciente a la imagen. Se puede establecer la forma en la que las coordenadas físicas de la localización de un punto en el espacio se transforman en las coordenadas en píxeles de su proyección en la imagen.

Sin embargo, al intentar la operación inversa se presenta un grave problema. A partir de una sola imagen de la escena no se puede averiguar sin ambigüedad a qué punto tridimensional corresponde cada píxel.

Por ello, se hace necesario emplear dos cámaras (o dos vistas) que permitan capturar simultáneamente imágenes de una misma escena según se muestra en la Figura 2.3.1, y así poder generar un mapa de profundidad que permita calcular con certeza las distancias a los objetos de interés.

Esto último es posible gracias a la geometría epipolar, geometría básica usada en visión estéreo, que proporciona las herramientas para relacionar dos cámaras entre sí y el mundo real.

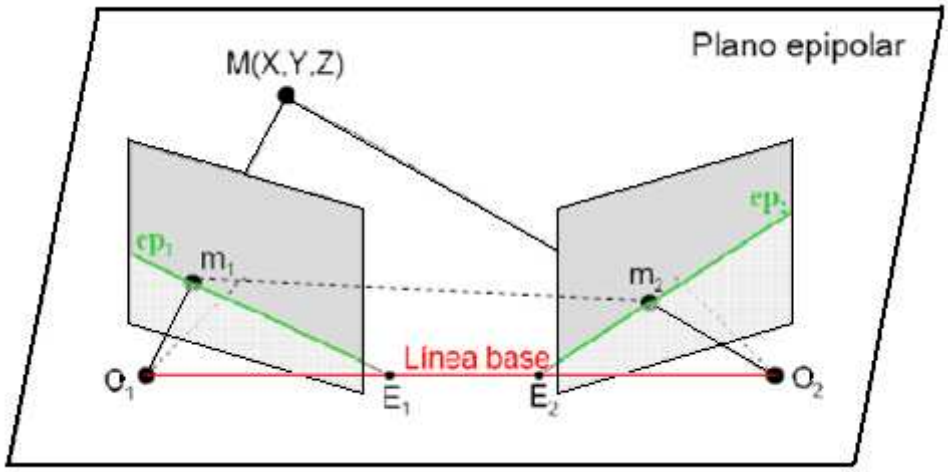


Figura 2.3.1: Esquema de relación de puntos en la geometría epipolar

Partiendo del esquema de la Figura 2.3.1 se considerará lo siguiente:

- M es un punto del espacio. Se proyecta en cada una de las cámaras izquierda y derecha, generando los puntos m1 y m2 respectivamente.
- **O_1** y **O_2** son los centros ópticos (o de proyección) de las cámaras.
- La rectas **ep_1 y ep_2** son las denominadas líneas epipolares.
- E1 y E2 son los denominados epipolos, que se definen como la proyección del centro óptico de una de las cámaras sobre el plano de proyección de la otra cámara. Además, se cumple que toda línea epipolar de una imagen, atraviesa el epipolo de la misma.

Observando la cámara izquierda según se muestra en la Figura2.3.2, se puede observar que cualquier punto situado en la recta **m_1O_1** tiene una misma proyección **m_1**. Por ello a partir de una única proyección **m_1** no es posible determinar a qué punto **M** corresponde la misma. A esta ambigüedad se le conoce como oclusión.

Si analizamos la Figura 2.3.2, vemos que con una sola imagen, los tres puntos representados en la escena aparecerán como un solo punto en la imagen capturada, debido a que se solapan entre ellos. Esta información de la profundidad de la escena la

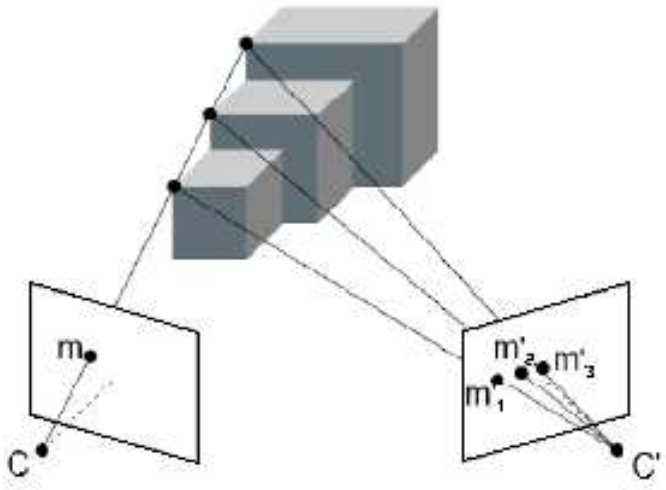


Figura 2.3.2: Representación de una oclusión, en la que se puede observar la necesidad de una segunda imagen para poder estimar la estructura 3D del objeto.

recuperamos mediante una segunda imagen, donde aparecen los tres puntos que perdíamos con la anterior imagen.

Para poder utilizar la geometría epipolar, y así poder triangular, se han de tener en cuenta la siguiente lista de afirmaciones:

- Cualquier punto **M** dentro del campo de visión de las cámaras se encuentra situado en el plano epipolar, por lo tanto, tendrá dos proyecciones (m_1, m_2).

- Un punto m_i situado en el plano proyectivo de una de las cámaras, tendrá una proyección asociada de la otra imagen, que se situará a lo largo de la línea epipolar de esa otra cámara. Esto establece una imposición llamada restricción epipolar, la cual dictamina que una vez conocida la geometría epipolar del montaje, es posible realizar una búsqueda bidimensional de los pares de proyecciones a lo largo de las líneas epipolares, lo que permitirá un ahorro computacional a la hora de realizar búsquedas de correspondencias, que son los pares de proyecciones asociadas a un mismo punto M del espacio 3D. Además ayudará a descartar falsos positivos más fácilmente.

- Cualquier plano epipolar asociado a un punto del espacio, intersecará siempre a la recta que une O_1 y O_2.

- Si dos puntos $\mathbf{M_1}$ y $\mathbf{M_2}$ son visibles por ambas cámaras y aparecen en el horizonte en una disposición determinada, entonces sus proyecciones en las imágenes guardarán el mismo orden.

Todos estos supuestos facilitarán el cálculo de correspondencias entre pares de imágenes y permitirán la triangulación de puntos en el espacio. En otras palabras, obtener la posición física de un punto en el espacio 3D.

2.4 La Matriz Esencial y la Matriz Fundamental

Las matrices que se van a describir a continuación, son unas matrices que contienen la información que relaciona las diferentes vistas o escenas que se obtienen al relacionar las escenas dos a dos.

- La matriz esencial *E* contiene información sobre la traslación y rotación que relaciona las dos cámaras físicamente en el espacio.

- La matriz fundamental *F*. Contiene toda la información que contiene la matriz esencial, con el añadido de que también posee en ella la información de los parámetros intrínsecos de la cámara.

2.4.1 Matriz Esencial

La matriz esencial E nos da la relación existente entre las coordenadas de un punto del espacio M vista desde cada una de las cámaras. Es decir, nos permite pasar de las coordenadas de una vista a las coordenadas de la otra vista mediante una rotación R y una traslación T. Este aspecto puede observarse en la Figura 2.4.1.

A continuación se procede a deducir matemáticamente la matriz esencial.

Si llamamos $\mathbf{M_L}$ y $\mathbf{M_R}$ a la posición tridimensional del punto **M** referida al origen de coordenadas de la cámara izquierda y derecha respectivamente. Además supóngase que el origen de coordenadas global del sistema es el centro de proyección $\mathbf{O_L}$ como se muestra en la Figura 2.4.1. En consecuencia:

- La posición del punto **M** respecto a ese origen de coordenadas global coincidirá con $\mathbf{M_L}$.

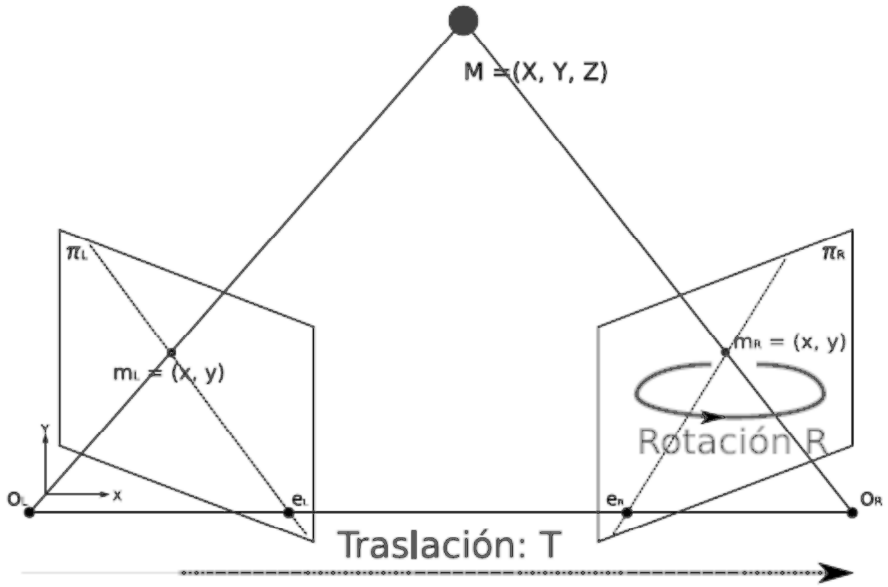


Figura 2.4.1: Efecto de aplicar la matriz esencial E a una vista. Se puede observar como la relación existente entre los dos puntos de vista, son una traslación T y una rotación R.

- El origen de coordenadas $\mathbf{O_R}$ estará a una distancia **T** de $\mathbf{O_L}$. **T** es el vector de translación.

- El punto **M** visto desde la cámara derecha podrá expresarse como

$$M_R = R(M_L - T) \qquad (2.36)$$

Sea un plano definido por un vector **n** normal, que atraviesa un punto **a** de ese plano. Entonces se cumplirá que todos los puntos **x** pertenecientes al mismo plano, obedecerán la siguiente restricción

$$(x - a)^T n = 0 \qquad (2.37)$$

Como $\mathbf{M_L}$ se encuentra dentro del mismo plano epipolar, y además $\mathbf{T \times M_L}$ es un vector ortogonal a $\mathbf{M_L}$, entonces se podrá aplicar la propiedad (2.37) de forma que

$$(M_L - T)^T (T x M_L) = 0 \qquad (2.38)$$

Obsérvese que se cumple $R^{-1} = R^T$, ya que cualquier matriz de rotación R es ortonormal y por tanto su traspuesta y su inversa son iguales. Entonces la ecuación (2.36) puede expresarse como $(\mathbf{M_L - T}) = \mathbf{R^{-1}} \cdot \mathbf{M_R}$, por lo que

$$(R_T M_R)^T (T x M_L) = M_R^T R (T x M_L) = 0 \qquad (2.39)$$

Además, es posible [3] rescribir el producto escalar **T x ML** como

$$T x M_L = S M_L \Rightarrow S = \begin{pmatrix} 0 & -T_x & T_y \\ T_x & 0 & -T_x \\ -T_y & T_x & 0 \end{pmatrix} \qquad (2.40)$$

así que la ecuación (2.39) se podrá expresar como

$$M_R^T R S M_L = 0. \qquad (2.41)$$

Considérense ahora las relaciones

$$m_L = \frac{f_L M_L}{Z_L} \qquad (2.42)$$

$$m_R = \frac{f_R M_R}{Z_R} \qquad (2.43)$$

Teniéndolas en cuenta, dividiendo la ecuación (3.6) por $\frac{Z_L Z_R}{f_L f_R}$ y definiendo $E = RS$ entonces

$$\mathrm{m}_{\mathrm{R}}^{\mathrm{T}} E \mathrm{m}_{\mathrm{L}} = 0 \qquad (2.44)$$

que es la expresión que relaciona los puntos de las imágenes de ambas cámaras. Esta relación se cumple gracias a la matriz esencial *E*, la cual tiene las siguientes características:

- Tiene dimensiones 3x3 y contiene cinco parámetros: tres que indican la rotación y dos que indican la dirección de la translación.

- Su determinante es cero, ya que es una matriz de rango 2. Como consecuencia, da lugar a un sistema lineal de ecuaciones que proporcionará más de una solución.

Esta matriz permite calcular una reconstrucción métrica de la escena, que quiere decir que la nube de puntos obtenida sólo es ambigua frente a transformaciones de escalado.

Conociendo la reconstrucción 3D y sus correspondientes proyecciones 2D se puede resolver la pose de la cámara para cada fotograma usando el algoritmo DLT (Direct Linear Transformation).

2.4.2 Matriz Fundamental

Como ya se ha expuesto previamente, la matriz *E* contiene información acerca de la geometría relativa a las dos cámaras. Sin embargo, no es capaz de proporcionar los parámetros intrínsecos de las cámaras del sistema y tampoco maneja las coordenadas de los puntos en píxeles. Esto último es necesario si se quiere relacionar los píxeles de una imagen con sus correspondientes líneas epipolares en la otra, de manera que se pueda facilitar la búsqueda de correspondencias. Es por ello que el cálculo y uso de la matriz fundamental *F* se vuelve indispensable.

Acto seguido se procede a derivar la matriz *F*.

Sea $\mathbf{m_L}$ un punto de la imagen izquierda situado sobre la línea epipolar L_L. Según se expuso en el apartado 2.3, toda línea epipolar pasa por el epipolo de la imagen en cuestión, por lo que L_L se podrá definir como

$$L_L = e_L x\, m_L = [e_L]_x m_L \qquad (2.45)$$

donde

$$[e_L]_x = \begin{pmatrix} 0 & -e_x & e_y \\ e_x & 0 & -e_x \\ -e_y & e_x & 0 \end{pmatrix} \qquad (2.46)$$

Teniendo en cuenta la ecuación

$$L_R \approx (P_R)^{-1} P_L^T L_L \qquad (2.47)$$

y usando además la expresión (2.45)

$$L_R \approx (P_R)^{-1} P_L^T [e_L]_x m_L \qquad (2.48)$$

donde la matriz fundamental es

$$F \approx P_R^{-1} P_L^T [e_L]_x \qquad (2.49)$$

El hecho de que $\mathbf{m_R}$ se encuentre a lo largo de la línea L_R se puede expresar así

$$m_R^T L_R = 0 \qquad (2.50)$$

que junto con las ecuaciones (2.48) y (2.49) permite llegar a la expresión que relaciona los puntos de ambas imágenes

$$m_R^T F m_L = 0 \qquad (2.51)$$

De todo lo anterior se deduce que dada la matriz *F* y un punto de una imagen $\mathbf{m_L}$ o $\mathbf{m_R}$ entonces

$$L_R \approx F m_L \qquad (2.52)$$

$$L_L \approx F m_R \qquad (2.53)$$

Finalmente, si se tiene en cuenta que $F \approx P_R^{-1} P_L^T [e_L]_x$ y que $[\mathrm{a}]_\mathrm{x}\mathrm{a} = 0$ para cualquier *a* entonces

$$F e_L = 0\,, \qquad (2.54)$$

restricción que se usa para calcular el epipolo de la imagen izquierda.

2.4.3 Cálculo de la Matriz Fundamental

Existen varios algoritmos que calculan la matriz *Fundamental*. Todos ellos requieren de un número de puntos de cada imagen para su correcta ejecución.

En el caso de estudio estos puntos son las esquinas detectadas en el tablero de ajedrez.

El algoritmo de 7-puntos usa exactamente 7 puntos de cada imagen en conjunción con el hecho de que la matriz *F* siempre es de rango 2. La desventaja de esta restricción es que provoca que hasta tres matrices *F* puedan darse como posible solución, y llegado el caso habría que realizar una discriminación adicional de los resultados.

El algoritmo de 8-puntos calcula *F* a partir de un sistema lineal de ecuaciones. Si se utilizan más de 8 puntos la solución será mejor, ya que se podrá aplicar un algoritmo de minimización del error. En cualquier caso, el problema de ambos algoritmos es que son muy sensibles respecto a los valores atípicos (esquinas del tablero cuya detección no ha sido suficientemente precisa), incluso si se usan más de 8 puntos.

Por otro lado, los algoritmos RANSAC (RAndom SAmpling Consensus) y LMedS (Least Median of Square) son más robustos, ya que tienen la capacidad de reconocer y eliminar los valores atípicos, por lo que la solución no se ve alterada debido a la presencia de éstos.

Para obtener resultados precisos con cualquiera de ellos es necesario disponer de más de 8 puntos.

De estos 4 algoritmos, el elegido para su implementación en este proyecto es el algoritmo RANSAC [6], ya que es muy robusto y uno de los más utilizados, lo que hace fácil su implementación.

En cada una de las iteraciones que RANSAC realiza, se toma un subconjunto de los puntos de la imagen de manera aleatoria y con ellos se calcula la matriz fundamental. Esta se va refinando en cada repetición, hasta que se cumple un criterio de calidad [9] que permita dar validez al resultado obtenido.

2.5 Rectificación de imágenes

La etapa de rectificación de las imágenes tiene gran importancia en el campo de la reconstrucción, ya que nos permite, aparte de eliminar las distorsiones provocadas por la cámara, alinear los pares de imágenes, haciendo coincidir sus líneas epipolares, con el fin de convertir la búsqueda de correspondencias a un problema unidimensional. En la Figura 2.5.1, se puede observar el efecto de la rectificación.

Consiguiendo esto, se consigue optimizar tanto los algoritmos de búsqueda, como el tiempo computacional, ya que se trata de realizar una búsqueda de dos dimensiones a una dimensión.

Al hacer uso de los algoritmos de calibración que a continuación se describen, se consigue también calcular la disparidad existente entre las dos imágenes, obteniendo así una idea de la profundidad a la que se encuentran los objetos (ver figura 2.5.2).

Figura 2.5.1: Par de imágenes rectificados. En ella se puede observar como todos los puntos de la imagen derecha y la izquierda, se encuentran sobre la misma horizontal.

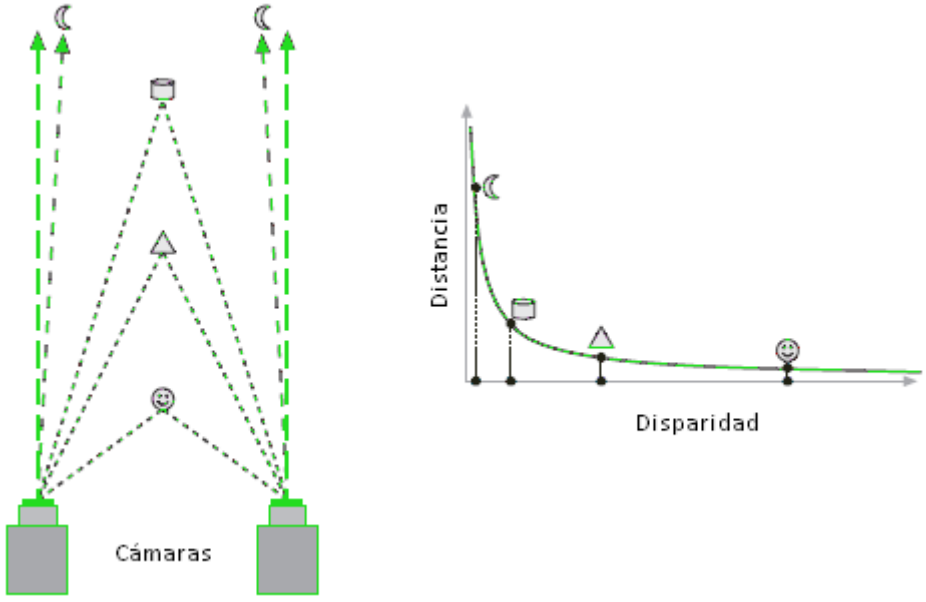


Figura 2.5.2: Esquema de efecto de la disparidad al rectificar las imágenes.

2.5.1 Algoritmo de Hartley

El algoritmo de Hartley trata de encontrar homografías que transformen las imágenes, de manera que sus epipolos se sitúen en el infinito. Para ello utiliza correspondencias entre puntos de pares de imágenes que o bien le son previamente dados, o bien es él mismo quien las busca en el conjunto de imágenes a rectificar.

Por tanto, Hartley es capaz de calcular la matriz *F* a partir de puntos que él mismo decida utilizar. Su mayor ventaja por tanto, es que es capaz de obtener la matriz *F* por sí mismo sin que se haya calculado previamente.

Entre las desventajas que caracterizan a este algoritmo, destaca el hecho de que no tiene en cuenta las matrices intrínsecas de las cámaras. Tampoco es capaz de calcular las matrices de proyección rectificadas, que son las matrices P_L y P_R que contienen los parámetros intrínsecos tal que las longitudes focales, centros ópticos y puntos principales son iguales en ambas cámaras. Esto se traduce en que Hartley no tiene sentido de la escala de los objetos que aparecen en la escena. Por ejemplo, supóngase un objeto en la escena. Dado el caso, no se sabría si mide 10 metros de altura y está lejos de las cámaras, o si por el contrario es una miniatura que tiene 10 centímetros de alto y está cerca de ambas. Este efecto se puede apreciar en las figuras 2.5.3 (a) y 2.5.3 (b), y es un grave problema si se desea hacer una estimación de las distancias de los puntos en el espacio.

Otra desventaja es que este algoritmo no es adecuado para trabajar con imágenes donde la distorsión sea elevada. Por ello es conveniente utilizar imágenes donde éstas hayan sido corregidas mediante una calibración de las cámaras. En consecuencia, aunque Hartley sea capaz de calcular *F* sin un proceso previo de calibración, esto casi nunca es recomendable y por tanto habrá que obtener los parámetros intrínsecos en la mayoría de las ocasiones.

A continuación se muestran los pasos [6] que sigue el algoritmo de rectificación de Hartley de forma resumida:

1. Identificación de una serie de correspondencias $x_L \leftrightarrow x_R$ en cada par de imágenes. Se necesitan al menos 7 pares, si bien es preferible tener más con el fin de ganar precisión.

2. Cálculo de la matriz fundamental de forma similar a lo explicado en el apartado 2.4.2. Esto no será necesario si *F* ha sido obtenida previamente.

3. Cálculo de los epipolos de las imágenes según las expresiones

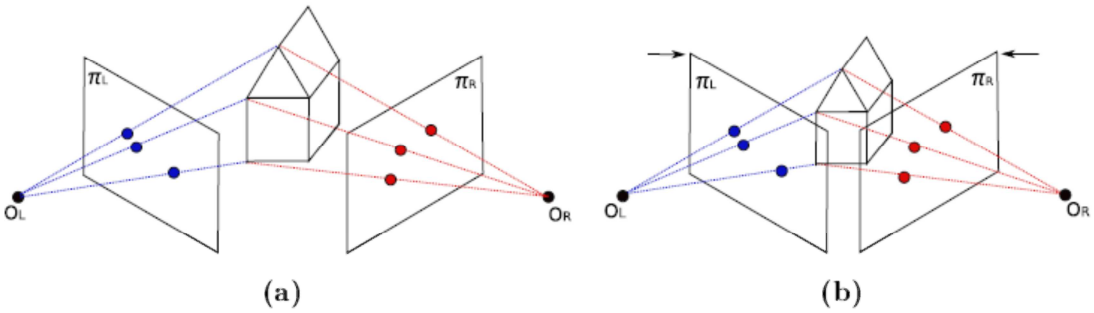


Figura 2.5.3: Se puede apreciar como el objeto de la Figura (a) tiene la misma proyección en (b) a pesar de haber sido escalada y acercada a las cámaras.

$$Fe_L = 0 \qquad (2.55)$$

$$(e_R)^t F = 0 \qquad (2.56)$$

que han sido derivadas a partir de lo expuesto en la sección 3.3.2.

4. Búsqueda de la homografía H_R que moverá el epipolo derecho e_R a un punto en el infinito. Esta homografía tendrá varios grados de libertad, lo que hará que algunas de las soluciones generen imágenes distorsionadas. Para que esto no ocurra se definirá dicha homografía en el entorno de un determinado punto x_0 como

$$H_R = GRT \qquad (2.57)$$

donde

$$G = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{-1}{f} & 0 & 1 \end{pmatrix} \qquad (2.58)$$

y donde los elementos de la expresión 2.57 son:

- T es el vector de traslación que moverá el vector x_0 hasta el origen del eje x.
- R es la matriz de rotación que trasladará el epipolo e_R al punto (f,0,1) en el eje x
- G es la matriz que se encarga de llevar el punto (f,0,1) al infinito.

En definitiva, esta homografía H_R funcionará para los puntos cercanos a un x_0 arbitrario, de manera que tan sólo se verán sometidos a una translación y rotación.

5. Encontrar la transformación *HL* para la imagen izquierda de forma que se minimice la suma distancias

$$\sum_i (H_L x_{Li}, H_R x_{Ri}) \qquad (2.59)$$

6. Finalmente se transforman las imágenes derecha e izquierda con las homografías H_R y H_L respectivamente, por lo que resultarán imágenes con sus filas de píxeles alineadas.

2.5.2 Algoritmo de Bouguet

El algoritmo de Bouguet necesita el vector **T** y la matriz de rotación ***R*** que relacionan las cámaras una respecto a la otra, por lo que es necesario haberlos obtenido previamente. Una primera ventaja de este método es que trata de que al transformar las imágenes, se muestre un elevado porcentaje de la parte relativa al solapamiento entre ambas cámaras.

Otra ventaja del algoritmo de Bouguet es que logra minimizar las distorsiones introducidas durante la rectificación. Esto lo consigue partiendo del hecho de que la rotación *R* de la imagen derecha equivale a dividir el giro completo en dos rotaciones r_L y r_R que se aplican a la imagen izquierda y derecha respectivamente. Con esto conseguiría que los rayos principales estuvieran paralelos (y por tanto los planos proyectivos coplanares). Sin embargo, no se lograría que las filas de píxeles quedaran

alineadas. Es por eso que lo que realmente hace, es calcular dos matrices R_L y R_R a partir de una matriz de rotación *Rrect* previamente rectificada.

Rrect es la matriz que trasladará el epipolo e_L al infinito y alineará las líneas epipolares a lo largo del eje *x*. Estando formada por tres vectores. El primero de ellos, será el vector unitario normalizado que indique la dirección del epipolo e_L, coincidiendo esta con el vector de translación T normalizado.

$$e_1 = \frac{T}{\|T\|} \tag{2.60}$$

El siguiente vector que conformará *Rrect* debe ser ortogonal a **e1**, ya que toda matriz de rotación está formada por vectores unitarios ortogonales entre sí. Una elección acertada es un vector que además sea ortogonal al rayo principal de la cámara izquierda. Así pues, haciendo el producto vectorial de la dirección del rayo principal y el vector *e*1, y normalizando, se obtiene

$$e_2 = \frac{(-T_y\ T_x\ 0)^t}{\sqrt{T_x^2+T_y^2}} \tag{2.61}$$

Y por último, el tercer vector, será e3 = e1x e2, de forma que la matriz *Rrect* quedará

$$R_{rect} = \begin{pmatrix} e_1^t \\ e_2^t \\ e_3^t \end{pmatrix} \tag{2.62}$$

por lo que

$$R_L = R_{rect} r_L \tag{2.63}$$

$$R_R = R_{rect} r_R \tag{2.64}$$

que serán las matrices de rotación rectificadas que girarán ambas cámaras, consiguiendo que sus ejes ópticos queden paralelos y las filas de píxeles de sus imágenes alineadas. Bouguet además calcula las matrices de proyección *PRrect* y *PLrect* rectificadas, y a partir de ellas construye la siguiente matriz

$$Q = \begin{pmatrix} 1 & 0 & 0 & -pL_x \\ 0 & 1 & 0 & -pL_y \\ 0 & 0 & 0 & f \\ 0 & 0 & \frac{-1}{T_x} & \frac{pL_x - pR_x}{T_x} \end{pmatrix} \quad (2.65)$$

donde

- pL_x y pR_x son las coordenadas del eje *x* de los puntos principales de las cámaras izquierda y derecha.

- f es la longitud focal de las cámaras rectificadas. Tendrá un único valor común a ambas.

- T_x es la componente *x* del vector de translación **T** que relaciona una cámara respecto a la otra.

La utilidad de esta matriz *Q* reside en el hecho de que permite proyectar puntos 2D de las imágenes al mundo real, consiguiendo establecer sin ambigüedad una correspondencia entre los puntos 3D y píxeles de las imágenes

$$Q \begin{pmatrix} x \\ y \\ d \\ 1 \end{pmatrix} = \begin{pmatrix} X \\ Y \\ Z \\ W \end{pmatrix} \quad (2.66)$$

donde

- (x,y) son las coordenadas euclídeas en píxeles del punto de la imagen

- (X,Y,Z,W) son las coordenadas homogéneas del punto 3D del mundo real. Por tanto, sus coordenadas euclídeas serán, (X/W,Y/W,Z/W).

- *d* representa la disparidad. Este parámetro se define básicamente como la diferencia de píxeles en una correspondencia de pares puntos de una imagen izquierda y derecha.

Finalmente se concluye que Bouguet es un algoritmo que permite rectificar las imágenes con menos distorsión que el algoritmo de Hartley. Además facilita una herramienta para obtener la localización 3D de los objetos de la escena, siendo necesario para ello obtener la disparidad de cada uno de los píxeles de las imágenes.

2.6 Búsqueda de correspondencias

Una vez que hemos obtenido los parámetros de la cámara que nos permiten modificar las imágenes para dejarlas libres de distorsiones y aberraciones producidas por las imperfecciones de la cámara, y tras realizar la rectificación de las imágenes tomadas por la cámara, el siguiente paso es el de la búsqueda de correspondencia entre las imágenes, con el fin de poder relacionar las imágenes entre sí y poder obtener parámetros como la pose de la cámara o la situación tridimensional de los puntos de las imágenes.

El problema al que nos enfrentamos sería el de detectar un numero 'x' de emparejamientos de puntos homólogos.

El primer paso en la búsqueda de correspondencias, es buscar los puntos de la imagen que aporten información realmente importante. En general, estos puntos, denominados de interés o característicos, tienen en común una serie de propiedades. Algunas de ellas son, por ejemplo, que se pueden encontrar de una manera sencilla y formalizable. Es decir, al someter la imagen a una operación algorítmica determinada, dichos puntos destacan significativamente. Además, tienen una posición muy bien definida y el conjunto de píxeles vecinos que hay alrededor del punto característico aporta una gran cantidad de información local relevante.

La propiedad más importante es su estabilidad frente a perturbaciones locales y globales. Éstas pueden incluir transformaciones simples como rotaciones y traslaciones, o bien variaciones más complejas como cambios de perspectiva, luminosidad y escala. Con el término 'cambio de escala' nos referimos a variaciones que se puedan dar en el tamaño de un objeto que aparezca en la imagen, debido por ejemplo, a un desplazamiento longitudinal de la cámara.

Para realizar la búsqueda de este conjunto de puntos de interés existen diferentes métodos. En función del tipo que busquemos utilizaremos una técnica u otra. Es decir, ciertos operadores aplicados sobre la imagen aportan información sobre píxeles con carácter de esquina, mientras que otros pueden dar información sobre bordes. Es importante definir previamente qué tipo de puntos se quieren detectar, ya que algunos serán más estables a determinadas transformaciones, mientras que otros serán invariantes a otro tipo de cambio.

El siguiente paso, una vez calculado dónde están las zonas de interés, es describir la zona en cuestión. Los puntos concretos que nos proporcionan cualquiera de los métodos anteriores nos aportan información de localización, sin embargo, no es suficiente para una posterior búsqueda de correspondencias entre imágenes. Por ese motivo, una vez los métodos anteriores nos dicen dónde se puede extraer información, hay que describir el vecindario del punto. Dicha descripción de los alrededores del punto se llevará a cabo en un radio determinado. Cuanto mayor sea este radio, mayor coste computacional conllevará su cálculo, pero se describirá con

mayor amplitud cada región de la imagen. Por el contrario, no interesan descriptores demasiado grandes, ya que en ese caso se perdería el concepto de 'localidad' en los descriptores. Por tanto, hay que llegar a un término medio que proporcione un algoritmo eficiente y a su vez que describa suficientemente cada zona importante.

Existen diferentes maneras de describir localmente los alrededores del punto característico. Una forma puede ser evaluando el nivel de gris de los píxeles vecinos. Partiendo de que la imagen está normalizada a la unidad, aquellos píxeles más oscuros tendrán un valor cercano a cero y los más claros serán próximos a uno. También cabe la posibilidad de evaluar el nivel de color. Para ello, hay que tener en cuenta que hay que analizar las tres matrices que definen el color: R, G y B. El color de una región específica puede ser representado mediante sus tres histogramas de color, o bien calculando la media de la región (y por tanto obteniendo tres escalares).

Dejando de lado los niveles tanto de gris como de color, una tercera opción es la de calcular y detallar las orientaciones de los gradientes alrededor del punto característico. De la misma forma que antes, se pueden estudiar mediante el uso de histogramas o el cálculo de medias, entre otros. Esta multitud de posibilidades en el momento de la descripción local de la imagen es independiente del método escogido a la hora de buscar los puntos de interés.

La descripción de los vecindarios nos permitirá asociar los puntos clave de una imagen de entrenamiento con otra de test, y por tanto seremos capaces de identificar puntos correspondientes entre ellas. Por ese motivo es tan importante que todos estos descriptores locales sean insensibles a cambios de escala, rotaciones, traslaciones, etc. Mientras en una imagen A (de entrenamiento) aparece un objeto en una posición determinada, en una imagen B (de test) puede aparecer el mismo objeto rotado o escalado de diferente forma. Sin embargo, el algoritmo debe ser invariante a esos cambios, gracias a las propiedades singulares de los puntos característicos y sus correspondientes descriptores.

Existen multitud de métodos para calcular estos puntos de interés. Los descriptores tienen que tener idealmente una serie de características:

- **Simplicidad**: El descriptor debería representar las características extraídas de la imagen de manera clara y sencilla para permitir una fácil interpretación de su contenido.

- **Repetibilidad**: El descriptor generado a partir de una imagen debe ser independiente del momento en el que se genere.

- **Diferenciabilidad**: Dada una imagen, el descriptor generado debe poseer alto grado de discriminación respecto de otras imágenes y al mismo tiempo contener información que permita establecer una relación entre imágenes similares.

- **Invarianza**: Cuando existen deformaciones en la representación de dos imágenes, es deseable que los descriptores que las representan aporten la robustez necesaria para poder relacionarlas aún bajo diferentes transformaciones.

- **Eficiencia**: Es deseable que los recursos consumidos para generar el descriptor sean aceptables para poder ser utilizados en aplicaciones con restricciones críticas de espacio y/o tiempo.

A la hora de escoger un método para calcular las correspondencias, cabe destacar que existen, fundamentalmente, dos tipos de descriptores de imagen:

- **Descriptores Globales**: Resumen el contenido de la imagen en un único vector o matriz de características. Poseen la ventaja de encapsular una gran cantidad de información de la imagen requiriendo una pequeña cantidad de datos para describirla. A pesar de su simplicidad, este tipo de descriptores han resultado ser ampliamente utilizados para diferentes tareas debido entre otras cosas a su bajo coste computacional unido a unas prestaciones relativamente buenas. Un representante de esta clase es el Histograma de Color.

- **Descriptores Locales**: Son utilizados en aquellas tareas en las que una descripción local del contenido de la imagen resulta más apropiado. Actúan sobre regiones de interés, previamente calculadas o identificadas, construyendo un vector de características de esa región que tiene en cuenta la información contenida tanto en el punto de interés como en la región adyacente al mismo o vecindario. Normalmente las regiones descritas se conocen como puntos de interés, también llamados puntos destacados o keypoints, sin embargo estas regiones suelen referirse a bordes o pequeñas partes de la imagen. El descriptor entonces, está constituido por la totalidad de los vectores de características calculados. A modo de ejemplo podríamos mencionar el descriptor local SIFT o el SURF.

En la Figura 2.6.1 se puede observar la diferencia principal entre estos dos tipos.

En la realización de este proyecto, nos basaremos en uno de los descriptores locales más utilizados en los últimos años, ya que cumple bastante bien las características que ha de tener un descriptor, así como una buena velocidad de cómputo. Nos referimos al descriptor SURF.

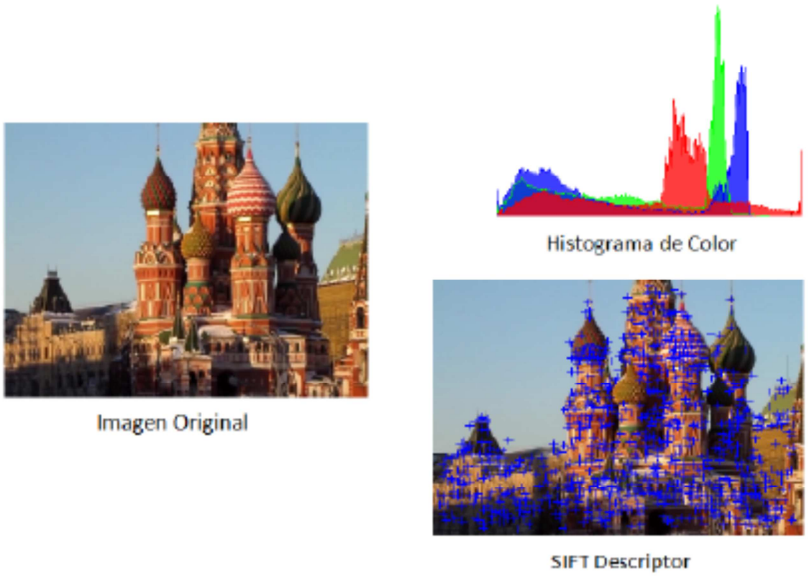


Figura 2.6.1: Descriptor local vs. Descriptor global

2.6.1 Descriptor SURF

El descriptor SURF, cuyo acrónimo significa Speeded-Up Robust Features, fue desarrollado por Herbert Bay [20] como un detector de puntos de interés y descriptor robusto. El descriptor SURF guarda cierta similitud con la filosofía del descriptor SIFT [17], si bien presenta notables diferencias que quedarán patentes con la siguiente exposición sobre su desarrollo. Los autores afirman sin embargo que este detector y descriptor presentan principalmente 2 mejoras resumidas en los siguientes conceptos:

- Velocidad de cálculo considerablemente superior sin ocasionar perdida del rendimiento.

- Mayor robustez ante posibles transformaciones de la imagen.

Estas mejoras se consiguen mediante la reducción de la dimensionalidad y complejidad en el cálculo de los vectores de características de los puntos de interés obtenidos, mientras continúan siendo suficientemente característicos e igualmente repetitivos.

A continuación se describirán en detalle las etapas para la creación de los descriptores SURF, si bien antes se presentan a modo de resumen previo las diferencias más originales respecto del descriptor SIFT:

- La normalización o longitud de los vectores de características de los puntos de interés es considerablemente menor, concretamente se trata de vectores con una dimensionalidad de 64, lo que supone una reducción de la mitad de la longitud del descriptor SIFT(dimensionalidad de 128).

- El descriptor SURF utiliza siempre la misma imagen, la original.

- Utiliza el determinante de la matriz Hessiana para calcular tanto la posición como la escala de los puntos de interés.

Debido al gran número de semejanzas entre estos dos descriptores, y a pesar de proceder a describir en detalle el funcionamiento del descriptor SURF, resulta interesante ir realizando algunas comparaciones entre ambos descriptores, por lo que se hará referencia en varias ocasiones durante la explicación del descriptor SURF a su semejante, el descriptor SIFT.

2.6.1.1 Detección de puntos de interés

La primera de las etapas tiene como objetivo obtener puntos candidatos de la imagen que puedan ser identificados de forma repetida bajo diferentes vistas del mismo objeto. El descriptor SURF es construido a partir del espacio-escala Gaussiano de la imagen original, en el cual se pueden detectar de manera efectiva las posiciones de los puntos claves, invariantes a cambios de escala de la imagen. El espacio-escala Gaussiano de una imagen *L*(x,y,σ) es definido como la convolución de funciones 2D Gaussianas *G*(x,y,σ) de diferentes valores σ con la imagen original *I*(x; y) :

$$L(x, y, \sigma) = G(x, y, \sigma) \cdot I(x, y) \qquad (2.67)$$

siendo (x,y) las coordenadas espaciales y σ el factor de escala. El algoritmo utiliza la función DoG (Diferencia de Gaussiana) que se forma a partir de la derivada escalar de la Gaussiana escalada espacialmente. Esta función DoG *D*(x, y, σ) se obtiene mediante la sustracción de escalas posteriores en cada octava:

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma) \qquad (2.68)$$

donde k es una constante multiplicativa del factor de escala. La función DoG es utilizada por varias razones. En primer lugar porque es una función eficiente en cuanto

a coste computacional se refiere: Las imagenes suavizadas $L(x,y,\sigma)$ son calculadas para la descripción de características en el espacio-escala, y por lo tanto, D puede obtenerse como una simple resta. Además, Mikolajczyk [16] asegura que los máximos y mínimos del Laplaciano de la Gaussiana respecto de una escala normalizada produce las características de imagen más en comparación con otras funciones como el Gradiente, el Hessiano o el detector de esquinas de Harris, pudiéndose aproximar el Laplaciano de la Gaussiana de escala normalizada mediante la función DoG.

Al conjunto de las imágenes Gaussianas suavizadas junto con las imágenes DoG se le llama octava. El conjunto de las octavas es construido mediante el muestreo sucesivo de la imagen original por un factor de 2. Cada una de las octavas es a su vez dividida en un número entero de subniveles o escalas *s*. Una vez se ha procesado una octava completa, la primera imagen de la siguiente octava se obtiene mediante el muestreo de la primera de las imágenes de la octava predecesora con un valor de σ del doble respecto a la actual. Esto se traduce en una gran eficiencia del algoritmo para un número de escalas pequeño. El proceso descrito puede verse representado en la Figura 2.6.2. Es importante tener en cuenta que la imagen original es expandida en el inicio del proceso para crear más puntos de muestreo que en la imagen original, por lo que la imagen resulta duplicada en tamaño antes de construir el primer nivel de la pirámide.

Dado que el espacio-escala $L(x,y,\sigma)$ representa la misma información a diferentes niveles de escala, el modo particular del muestreo permite una reducción de la redundancia. De esta manera se producen s + 3 imágenes por cada una de las octavas y por lo tanto s + 2 DoG imágenes donde se llevará a cabo la búsqueda de extremos.

Para detectar los máximos y los mínimos locales de cada punto de la imagen $D(x,y,\sigma)$ se compara el valor de éste con el de los puntos vecinos, en concreto, con el de sus 8 vecinos más próximos de la imagen D donde se encuentra el punto más los 9 vecinos de cada una de las imágenes D de nivel superior e inferior como se muestra en la Figura 2.6.3. Si el valor resulta ser superior o inferior al de todos sus vecinos, se identifica el punto como máximo o mínimo local respectivamente.

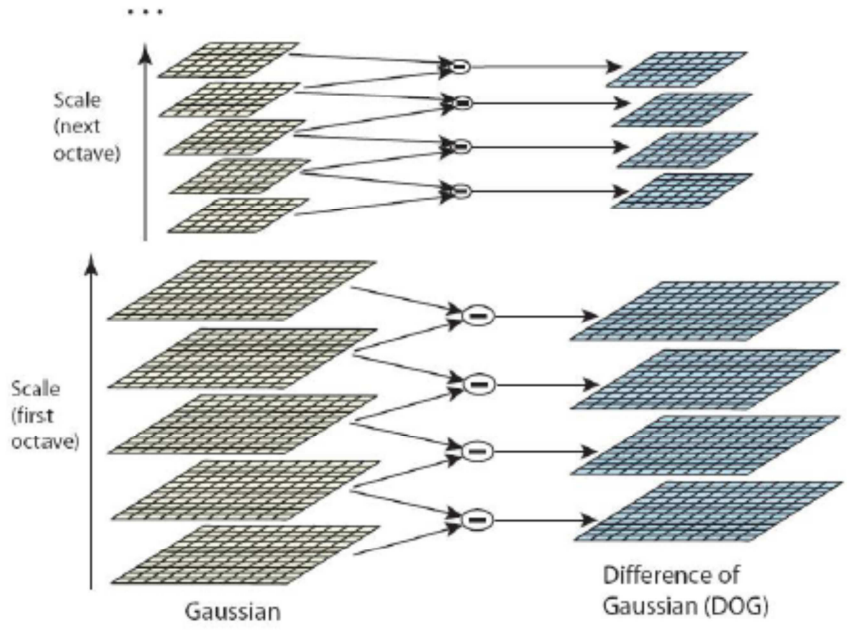


Figura 2.6.2: Creación del espacio-escala Gaussiano.
En cada una de las escalas, también llamadas octavas, la imagen se convoluciona repetidamente con funciones gaussianas para producir el conjunto de imágenes gaussianas mostradas en la parte izquierda de la imagen. Las imágenes obtenidas son substraídas en parejas adyacentes para producir las imágenes diferencia de gaussiana mostradas a la derecha. Después de cada octava, las imágenes Gaussianas son muestreadas por un factor de 2, y se repite el proceso.

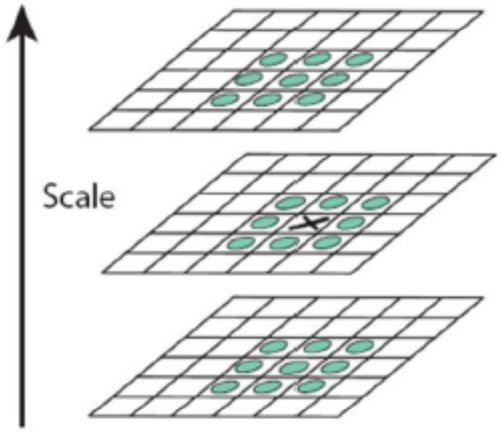


Figura 2.6.3: Localización de máximos y mínimos locales
Los máximos y mínimos de las imágenes diferencia-de-gaussiana son detectados mediante la comparación de un pixel (marcado con X) con sus 26 vecinos en las regiones de 3x3 de las escalas actual y adyacentes (marcados en azul).

La primera de las etapas del descriptor SURF es análoga a la del descriptor SIFT en cuanto a la detección de puntos de interés se refiere, si bien el procedimiento para su obtención se basa en diferencias sustanciales que se detallan a continuación.

El descriptor SURF hace uso de la matriz Hessiana, más concretamente, del valor del determinante de la matriz, para la localización y la escala de los puntos. El motivo para la utilización de la matriz Hessiana es respaldado por su rendimiento en cuanto a la velocidad de cálculo y a la precisión. Lo realmente novedoso del detector incluido en el descriptor SURF respecto de otros detectores es que no utiliza diferentes medidas para el cálculo de la posición y la escala de los puntos de interés individualmente, sino que utiliza el valor del determinante de la matriz Hessiana en ambos casos. Por lo tanto dado un punto *p* = (x, y) de la imagen *I*, la matriz Hessiana *H* (p,σ) del punto *p* perteneciente a la escala σ se define como:

$$H(p,q) = \begin{bmatrix} L_{xx}(p,q) & L_{xy}(p,q) \\ L_{xy}(p,q) & L_{yy}(p,q) \end{bmatrix} \qquad (2.69)$$

donde L_{xx} (p,σ) representa la convolución de la derivada parcial de segundo orden de la Gaussiana $\frac{\partial^2}{\partial x^2} g(\sigma)$ con la imagen *I* en el punto *p*. De manera análoga ocurre con los términos L_{xy} (p,σ), L_{yy} (p,σ) de la matriz.

A pesar de que los filtros gaussianos son óptimos para el análisis del espacioescala , se ha implementado una alternativa a los filtros gaussianos en el detector SURF debido a una serie de limitaciones de estos filtros (como la necesidad de ser discretizados, la falta de prevención total del indeseado efecto aliasing, etc.): los filtros tipo caja (de sus siglas en inglés box-filters). Estos nuevos filtros aproximan las derivadas parciales de segundo orden de las gaussianas y pueden ser evaluados de manera muy rápida usando imágenes integrales, independientemente del tamaño de éstas . Las imágenes integrales son calculadas mediante la siguiente fórmula:

$$I_{i\sum(x,y)} = \sum_{i=1}^{i\leq x} \sum_{j=1}^{j\leq y} I(i,j) \qquad (2.70)$$

donde (x, y) representan la posición del punto en la imagen y I_i (x, y)representa la intensidad de la imagen en el punto. Una vez la imagen integral ha sido creada, se puede calcular la suma de las intensidades de una región mediante una simple operación, como se puede observar en la Figura 2.6.4:

$$\sum I = I_{iA} + I_{iB} + I_{iC} + I_{iD} \qquad (2.71)$$

De esta forma, el tiempo necesario para el cálculo de las operaciones de convolución es independiente del tamaño de la imagen.

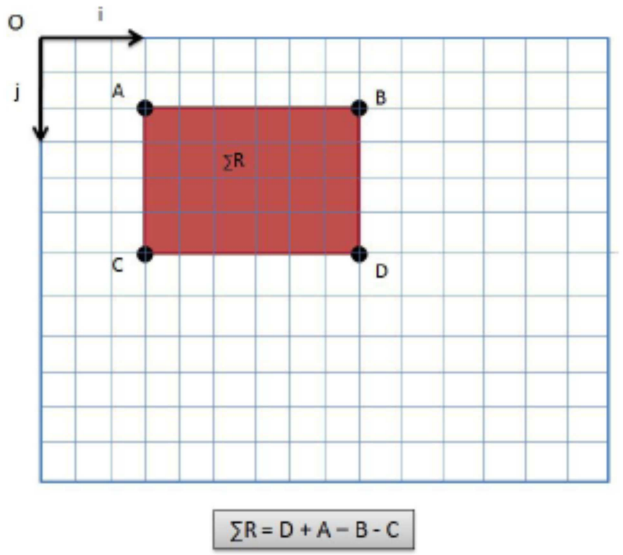


Figura 2.6.4: Representación de la intensidad de una región respecto de la imagen integral.

El espacio escala del descriptor SIFT, se crea a partir de imágenes suavizadas repetidamente mediante la aplicación de un filtro gaussiano y que posteriormente se submuestrean para alcanzar un nivel más alto dentro de la pirámide de dicho espacio escala. Sin embargo en el caso del detector SURF, debido a la utilización de filtros de tipo caja e imágenes integrales, no es necesario aplicar el mismo filtro iterativamente a la salida de una capa filtrada previamente, sino que se pueden aplicar dichos filtros de cualquier tamaño a la misma velocidad directamente sobre la imagen original. De este modo resulta que el espacio escala es analizado mediante la elevación del tamaño del filtro, en vez de reducir el tamaño de la imagen como es el caso del detector SIFT como se puede observar en la Figura 2.6.5.

Las aproximaciones de las derivadas parciales se denotan como D_{xx}, D_{xy}, y D_{yy}. En cuanto al determinante de la matriz Hessiana, éste queda definido de la siguiente manera:

$$\det(H_{aprox}) = D_{xx}D_{xy} - (0{,}9D_{xy})^2 \qquad (2.72)$$

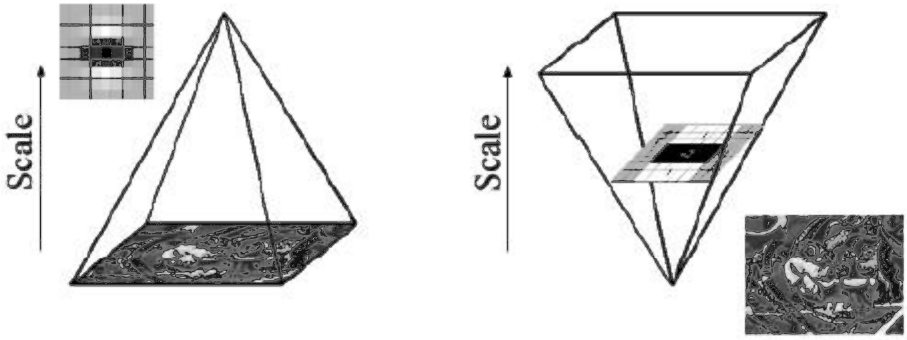


Figura 2.6.5: Espacio escala SIFT vs. SURF

donde el valor de 0,9 está relacionado con la aproximación del filtro gaussiano. En la Figura 2.6.6 se puede observar la representación de la derivada parcial de segundo orden de un filtro gaussiano discretizado y la aproximación de la derivada implementada en el caso del descriptor SURF.

Derivada parcial de segundo orden discretizadas en la dirección xy e y, L_{xy} y L_{yy} respectivamente (arriba), y sus respectivas aproximaciones D_{xy} y D_{yy} (abajo).

La imagen de salida obtenida tras la convolución de la imagen original con un filtro de dimensiones 9 x 9, que corresponde a la derivada parcial de segundo orden de una gaussiana con σ = 1,2, es considerada como la escala inicial o también como la máxima resolución espacial (s = 1,2, correspondiente a una gaussiana con σ = 1,2). Las capas sucesivas se obtienen mediante la aplicación gradual de filtros de mayores dimensiones, evitando así los efectos de aliasing en la imagen.

El espacio escala para el descriptor SURF, al igual que en el caso del descriptor SIFT, está divido en octavas. Sin embargo, en el descriptor SURF, las octavas están compuestas por un número fijo de imágenes como resultado de la convolución de la misma imagen original con una serie de filtros cada más grande. El incremento o paso de los filtros dentro de una misma octava es el doble respecto del paso de la octava anterior, al mismo tiempo que el primero de los filtros de cada octava es el segundo de la octava predecesora. De esta manera obtenemos las siguientes series de octavas con sus respectivos filtros como se muestra en la Figura 2.6.7:

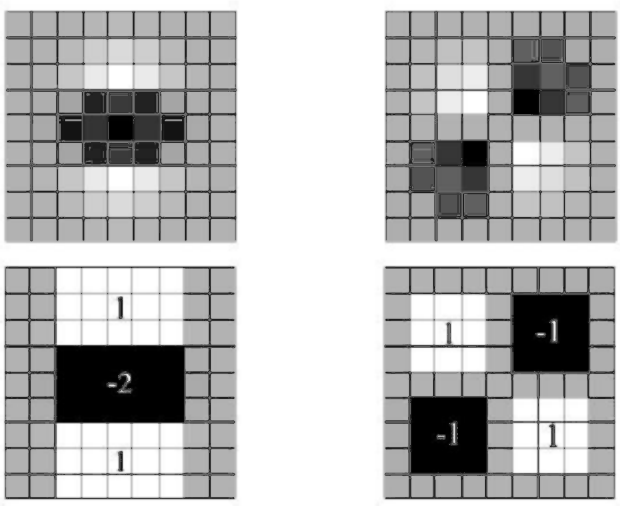


Figura 2.6.6: Derivadas parciales de segundo orden de un filtro gaussiano y su aproximación.

- Octava inicial: $9x9 \overset{6}{\rightarrow} 15x15 \overset{6}{\rightarrow} 21x21 \overset{6}{\rightarrow} 27x27$
- Octava siguiente: $15x15 \overset{6}{\rightarrow} 27x27 \overset{6}{\rightarrow} 39x39 \overset{6}{\rightarrow} 51x51$
- Octava siguiente: $27x27 \overset{6}{\rightarrow} 51x51 \overset{6}{\rightarrow} 75x75 \overset{6}{\rightarrow} 99x99$
- Y así sucesivamente …

Finalmente para calcular la localización de todos los puntos de interés en todas las escalas, se procede mediante la eliminación de los puntos que no cumplan la condición de máximo en un vecindario de 3 x 3 x 3. De esta manera, el máximo determinante de la matriz Hessiana es interpolado en la escala y posición de la imagen. En este punto se da por concluida la etapa de detección de los puntos de interés.

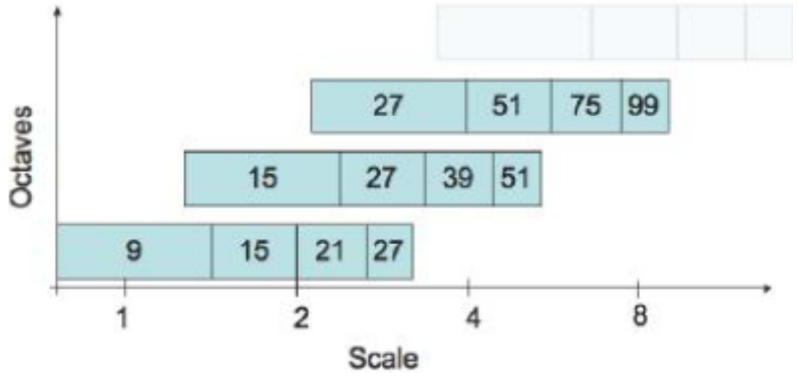


Figura 2.6.7: Representación gráfica de la longitud de los filtros de diferentes octavas.

2.6.1.2 Asignación de la orientación

La siguiente etapa en la creación del descriptor corresponde a la asignación de la orientación de cada uno de los puntos de interés obtenidos en la etapa anterior. Es en esta etapa donde se otorga al descriptor de cada punto la invarianza ante la rotación mediante la orientación del mismo.

El primer paso para otorgar la mencionada orientación consiste en el cálculo de la respuesta de Haar en ambas direcciones x e y mediante las funciones representadas en la Figura 2.6.8. El área de interés para el cálculo es el área circular centrada en el punto de interés y de radio 6s, siendo s la escala en la que el punto de interés ha sido detectado. De la misma manera, la etapa de muestreo depende de la escala y se toma como valor s. Respecto de las funciones onduladas de Haar, se toma el valor 4s, por tanto dependiente también de la escala, como referencia, donde a mayor valor de escala mayor es la dimensión de las funciones onduladas.

Tras haber realizado todos estos cálculos, se utilizan imágenes integrales nuevamente para proceder al filtrado mediante las máscaras de Haar y obtener así las respuestas en ambas direcciones. Son necesarias únicamente 6 operaciones para obtener la respuesta en la dirección x e y. Una vez que las respuestas onduladas han sido calculadas, son ponderadas por una gaussiana de valor $\sigma = 2{,}5s$ centrada en el punto de interés. Las respuestas son representadas como vectores en el espacio colocando la respuesta horizontal y vertical en el eje de abscisas y ordenadas respectivamente. Finalmente, se obtiene una orientación dominante por cada sector mediante la suma de todas las respuestas dentro de una ventana de orientación móvil cubriendo un ángulo de $\pi/3$ siguiendo las especificaciones recomendadas por el autor [20]. La representación de este puede observarse en la Figura 2.6.9.

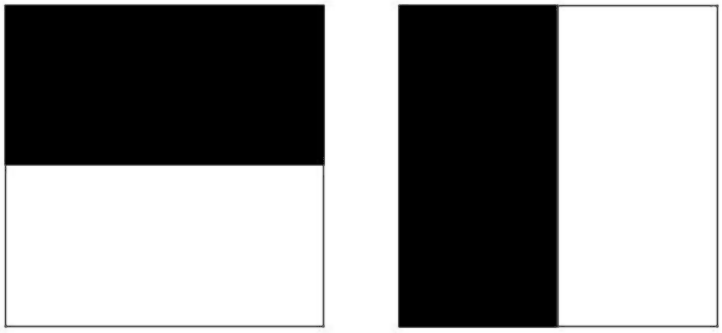


Figura 2.6.8: Filtros de Haar empleados en el descriptor SURF
Funciones de Haar para el cálculo de las respuestas en la dirección x (izquierda) e y (derecha). El color negro identifica el valor -1 y el color blanco el valor +1.

La orientación final del punto de interés será finalmente aquella cuyo vector sea el más grande dentro de los 6 sectores en los que han sido divididas el área circular alrededor del punto de interés.

2.6.1.3 Creación del descriptor

Es en esta última etapa del proceso donde se concreta la creación del descriptor SURF.

Se construye como primer paso una región cuadrada de tamaño 20s alrededor del punto de interés y orientada en relación a la orientación calculada en la etapa anterior. Esta región es a su vez dividida en 4 x 4 sub-regiones dentro de cada una de las cuales se calculan las respuestas de Haar de puntos con una separación de muestreo de 5 x 5 en ambas direcciones. Por simplicidad, se consideran d_x y d_y las respuestas de Haar en las direcciones horizontal y vertical respectivamente relativas a la orientación del punto de interés.

En la Figura 2.6.10 están representadas tanto las respuestas de Haar en cada una de las sub-regiones como las componentes d_x y d_y uno de los vectores.

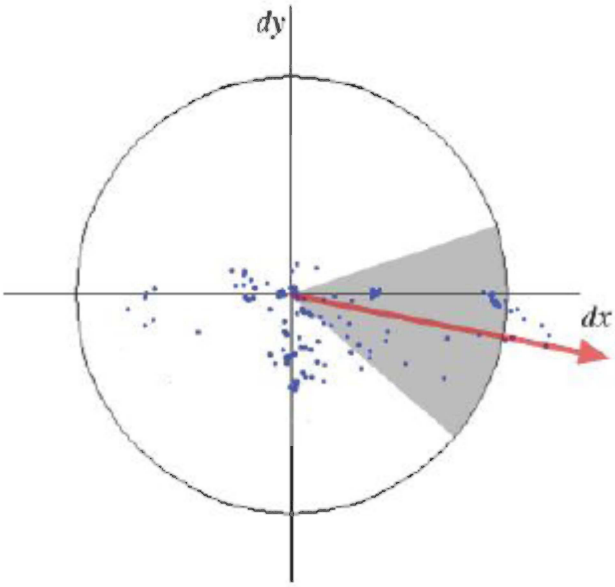


Figura 2.6.9: Asignación de la orientación de cada sector
Un vector de orientación es calculado como representante de todas las respuestas gaussianas de Haar en los puntos de muestreo contenidos en cada sector circular de valor π/3.

Para dotar a las respuestas d_x y d_y de una mayor robustez ante deformaciones geométricas y errores de posición, éstas son ponderadas por una gaussiana de valor σ=3,3s centrada en el punto de interés.

En cada una de las sub-regiones se suman las respuestas d_x y d_y obteniendo así un valor de d_x y d_y representativo por cada una de las sub-regiones. Al mismo tiempo se realiza la suma de los valores absolutos de las respuestas $|d_x|$ y $|d_y|$ en cada una de las sub-regiones, obteniendo de esta manera, información de la polaridad sobre los cambios de intensidad.

En resumen, cada una de las sub-regiones queda representada por un vector *v* de componentes:

$$v = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|) \qquad (2.73)$$

y por lo tanto, englobando las 4 x 4 sub-regiones, resulta un descriptor SURF con una longitud de 64 valores para cada uno de los puntos de interés identificados.

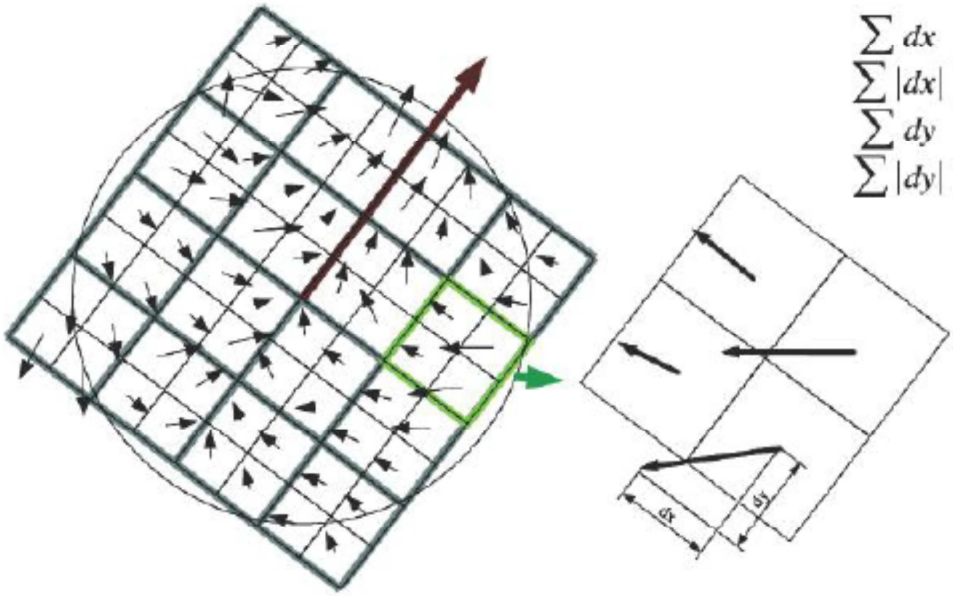


Figura 2.6.10: Respuestas de Haar en las sub-regiones alrededor del punto de interés

2.6.1.4 Matching entre puntos clave

El término matching entre imágenes tiene como finalidad el cálculo de un valor que represente el grado de similitud entre las dos imágenes, y que a continuación se puedan establecer las diferentes conclusiones. El cálculo de este valor, representado como distancia y conocido también como score, se realiza mediante la aplicación de una métrica o fórmula de la distancia entre ambas imágenes. Previo paso del cálculo del score, es necesario establecer las correspondencias entre los puntos clave.

La correspondencia entre puntos clave se lleva a cabo mediante el cálculo de la distancia euclídea entre los vectores de características pertenecientes a diferentes puntos de interés. Este cálculo genera a su vez otro valor que será utilizado para determinar cuál de los puntos de la imagen comparada se corresponde con su homólogo, en el caso de existir, de la primera de las imágenes.

Supongamos que queremos realizar el matching de puntos entre dos imágenes I1 e I2. Para cada uno de los puntos clave pertenecientes a I1, se seleccionan los dos mejores candidatos de entre todos los puntos clave pertenecientes a I2 mediante el criterio de máxima similitud. Este criterio establece que los mejores candidatos para realizar el matching con el punto clave p1 perteneciente a I1 cuyo vector de características es v1, son los puntos clave p01 y p02 pertenecientes a I2 cuyos vectores de características v01 y v02 representan las distancias euclídeas mínimas d1 y d2

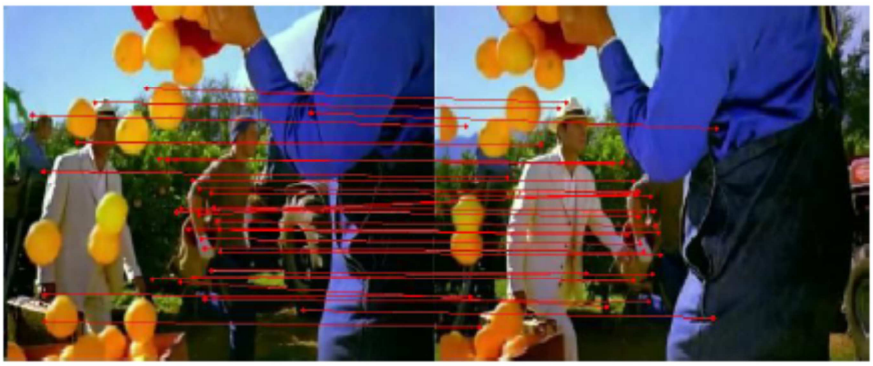


Figura 2.6.11: Representación del Matching del descriptor SURF

respectivamente respecto de v1. Si la relación d1=d2 entre las distancias mencionadas es suficientemente pequeña, entonces se establece el matching entre los puntos p1 y p01 pertenecientes a cada una de las imágenes. De acuerdo con [20], se establece un umbral de 0,7 para el ratio d1=d2.

Esta estrategia de matching recibe el nombre de “el vecino más próximo”. Finalmente la puntuación o score entre las dos imágenes se obtiene mediante una relación que tiene en cuenta el número total de puntos matcheados entre ambas imágenes.

En la Figura 2.6.11 se representa los resultados del matching de puntos clave entre dos imágenes para el descriptor SURF.

2.7 Mapas de disparidad y triangulación

Una vez tenemos las imágenes rectificadas, disponiendo de un problema unidimensional, el paso siguiente es obtener la profundidad de los puntos.

Existen diversas maneras de obtener la profundidad, como la triangulación (la cual se comentará más adelante) o la obtención de mapas de disparidad (la cual nos interesa en este apartado).

2.7.1 Mapas de disparidad

La disparidad se define como la diferencia en las coordenadas horizontales de los puntos p_L y p_R, o sea, d = x_L-x_R. Dependiendo el sistema de referencia utilizado en las imágenes, la definición puede cambiar de forma que el signo sea siempre positivo. Las coordenadas de p_L y p_R quedan relacionadas mediante:

$$\begin{Bmatrix} x_L = x_R + d \\ y_L = y_R \end{Bmatrix} \quad (2.74)$$

Como se puede apreciar en la figura 2.7.1 a medida que los puntos se encuentran más alejados de las cámaras, menor es su disparidad.

Una vez tenemos todo esto, podemos obtener las coordenadas 3D de los puntos mediante la expresión:

$$Z = \frac{f\,T}{d} \quad (2.75)$$

donde

- f es la distancia focal en píxeles de las cámaras. Recuérdese que tras la rectificación de las imágenes $f = f_R = f_L$.
- T es la distancia en unidades de longitud que separa los centros ópticos de las cámaras.
- d es la disparidad.
- Z es la distancia a la que se encuentra el punto 3D que generó las proyecciones de las imágenes cuya disparidad es *d*. Tiene las mismas unidades de longitud que *T*.

Esta expresión se deduce a partir de la relación de semejanza de triángulos existente en la Figura 2.7.1. Por tanto sólo se cumplirá si las imágenes han sido correctamente rectificadas.

En la figura 2.7.2 se puede observar un ejemplo de un mapa de disparidad.

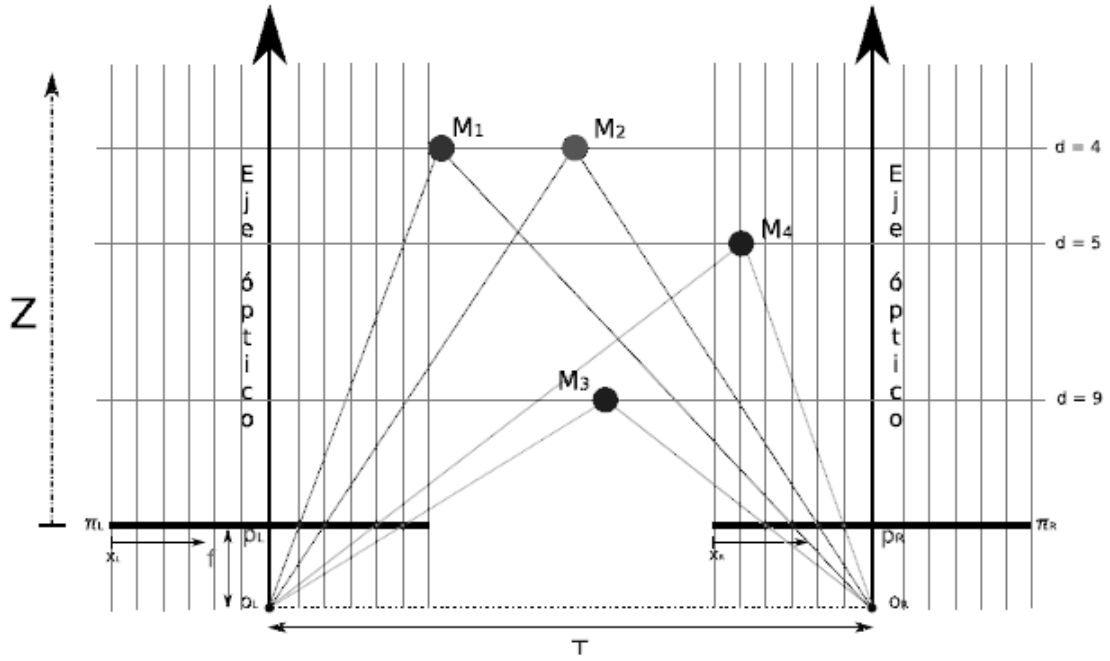


Figura 2.7.1: Esquema de disparidades para diferentes puntos

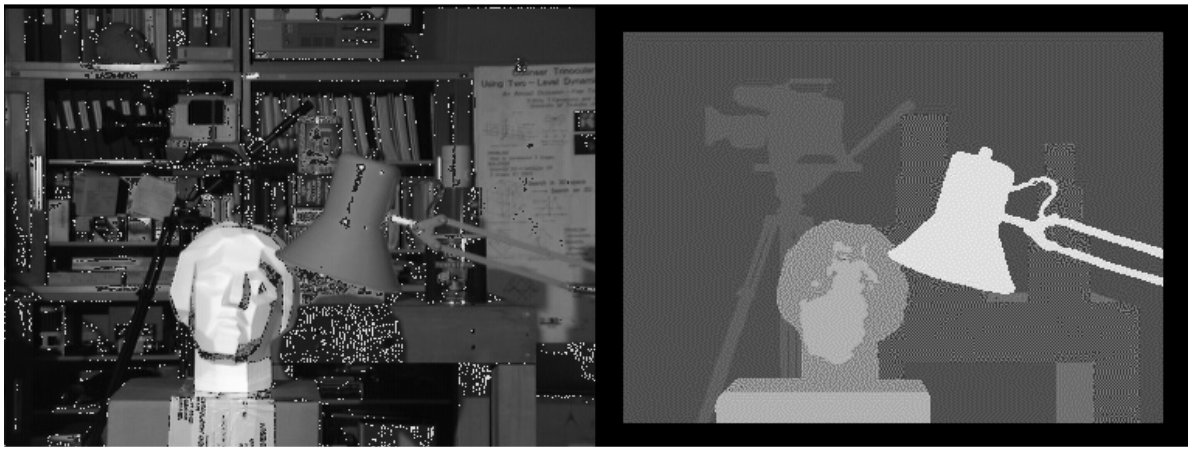


Figura 2.7.2: Ejemplo de mapa de disparidad (derecha) e imagen original (izquierda)

2.7.2 Triangulación

Suponiendo que un punto x en R^3, es visible en dos o más imágenes, llamando u y u' a las proyecciones del punto x en sendas imágenes, y suponiendo conocidas las matrices de proyección de las cámaras P y P', se puede calcular de manera trivial (en ausencia de ruido), la posición exacta del punto mediante la intersección de dos rayos que pasan por los puntos de proyección. A esto se le denomina triangulación.

En la vida real, la existencia de ruido se hace prácticamente inevitable, por lo que lo más probable es que los rayos no se corten, siendo necesario utilizar un algoritmo que calcule el punto de intersección que minimice el error de la medida.

El objetivo principal del algoritmo de reconstrucción, es , por tanto, minimizar la suma de los errores cuadráticos entre la posición de los valores medidos y los predichos del punto 3D en todas las imágenes en las que es visible dicho punto. Este se representa por:

$$X = min \sum_i \|u_i - \hat{u}_i(P_i, X)\|^2 \qquad (2.76)$$

donde u_i y $\hat{u}_i(P_i, X)$ son las posiciones medidas y predichas en la vista *i* (ver Figura2.7.3).

Para el caso de más de dos vistas, el problema de minimización puede conseguirse iterativamente mediante optimización no lineal.

Una opción muy común es la de partir de la asunción de que $u \approx PX$ ya que los vectores $u_i \; y \; los \; P_i X$ son paralelos, por lo que se puede escribir que:

$$[\tilde{u}_i]_x P_i \tilde{X} = 0 \qquad (2.77)$$

Esta ecuación tiene tres filas, pero proporcionas solamente dos restricciones sobre $\tilde{X}$ ya que cada fila puede ser expresada como combinación lineal de las otras dos. Todas estas restricciones puede ser reorganizadas en forma de matriz de la forma:

$$A\tilde{X} = 0 \qquad (2.78)$$

donde A es una matriz 3x4y n el número de vistas en la que los puntos reconstruidos son visibles.

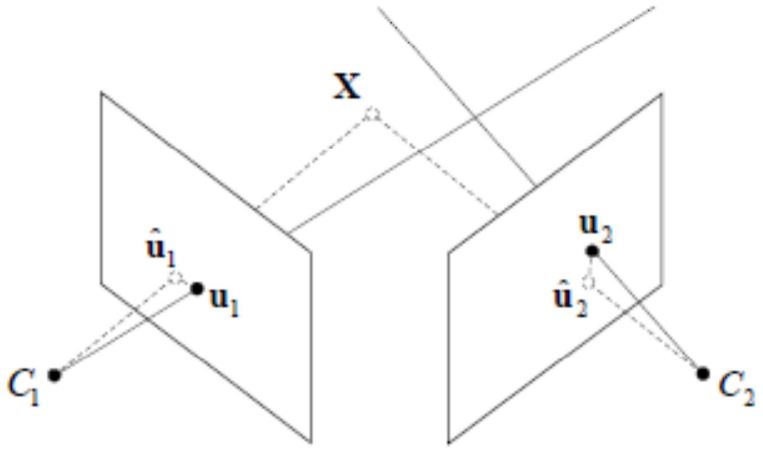


Figura 2.7.3 : Método de triangulación con existencia de ruido. Los rayos de ambas proyecciones no se cortan, por lo que se busca el valor que minimiza el error de medida, obteniendo el punto X.

La solución que se busca puede ser calculada mediante la resolución del problema de valores singulares de la matriz simétrica A^tA.

Para una información más detallada de triangulación y los diferentes métodos pueden consultarse en [21].

2.8 Structure From Motion a partir de múltiples vistas

Hasta ahora hemos podido sacar la conclusión de que las matrices Esencial o Fundamental, incorporan las restricciones geométricas relacionando cada par de imágenes. El siguiente paso es centrar la atención en reconstruir el problema de la estructura y el movimiento de un número aleatorio de imágenes.

Para que el último paso (Bundle adjustment) sea satisfactorio, es necesario tener un correcto sistema de inicialización, ya que en el caso contrario, el método podría fallar por la convergencia a un mínimo local de la solución.

Esta sección se basa en observar los algoritmos secuenciales y de factorización para el SFM de múltiples vistas.

2.8.1 Métodos Secuenciales

Estos métodos se basan en ir incorporando vistas sucesivas cada vez, como puede observarse en la Figura 2.8.1. Cada vista que se va introduciendo va obteniendo una reconstrucción parcial que se va añadiendo a la información que ya ha sido obtenida, para así ir obteniendo una reconstrucción más detallada y amplia de la escena.

Una inicialización aceptable es la obtenida por la descomposición de la matriz fundamental, relacionando las dos primeras vistas de la secuencia.

Existen varias estrategias para obtener el método de vistas sucesivas:

- Restricción epipolar: Una posibilidad es tener en cuenta la geometría epipolar que relaciona cada vista con su predecesora. Por ejemplo, cuando los parámetros intrínsecos de la cámara son conocidos, puede utilizarse la matriz esencial para obtener la posición relativa de la cámara (rotación y traslación).

- Unión de reconstrucciones parciales: Utilizando las correspondencias 3D es posible realizar uniones de vistas parciales. Normalmente, la reconstrucción de dos o tres vistas se puede realizar mediante la unión de los puntos 3D de varias imágenes adyacentes.

Los métodos secuenciales tienen importantes limitaciones. Las más importantes son que debe existir un gran número de puntos de interés en cada imagen, para así poder realizar una reconstrucción suficientemente buena, por otro lado, otra importante limitación es la necesidad de que exista un elevado grado de solapamiento entre imágenes consecutivas, lo que resulta prohibitivo computacionalmente para un gran número de vistas de una secuencia.

2.8.2 Métodos basados en factorización

A diferencia de los métodos anteriores, estos métodos trabajan calculando la pose y la geometría de la escena, utilizando todas las medidas de las imágenes simultáneamente. Una ventaja de este método es que los errores de reconstrucción asociados al cálculo de las coordenadas, puede distribuirse a lo largo de todas las medidas, por lo que los errores asociados al cierre de la escena pueden ser eliminados.

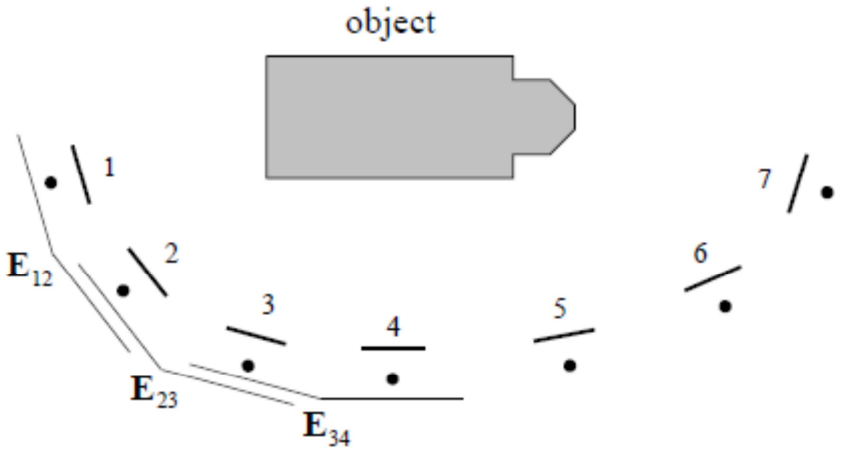


Figura 2.8.1: Esquema del método secuencial de representación en el que se van obteniendo las matrices esenciales comparando cada imagen con su predecesora.

Estos métodos no se suelen utilizar tan ampliamente, ya que al ser métodos iterativos, no se puede garantizar que su solución converja a la solución óptima.

Las limitaciones de estos algoritmos son más difíciles de salvar, ya que por ejemplo no pueden existir puntos ocultos, porque se requiere que todos los puntos sean visibles en todas las imágenes para realizar correctamente la factorización.

Para la realización de este proyecto se ha decidido utilizar un método secuencial ya que resulta un método más sencillo a la hora de observar la evolución de los resultados.

2.9 Open CV

OpenCV es un conjunto de bibliotecas de funciones de programación cuyo uso principal es la Visión por Computador en tiempo real. Su nombre proviene de los términos anglosajones Open Source Computer Vision [3]. Sus bibliotecas son de código abierto y desarrolladas en C/C++. Es multiplataforma y se puede ejecutar bajo Mac OS X, Windows, GNU/Linux, etc. Está diseñado para que pueda ser empleado conjuntamente con la biblioteca de Procesamiento de Imágenes de Intel (IPL).

Algunos de los recursos y operaciones que permiten las bibliotecas de OpenCV son:

- Procesado de imágenes y análisis de la misma.
- Capacidad de procesado desde diferentes fuentes de imagen o vídeo.
- Operaciones con vectores y matrices.
- Operaciones básicas y funciones de álgebra lineal.
- Estructuras de datos dinámicas como listas, colas y árboles.
- Procesamiento básico de imágenes entre los que se pueden destacar: filtros, detección de esquinas, conversiones de color, operaciones basadas en morfologías, histogramas y pirámide de imágenes.
- Análisis estructural, como análisis de contornos, transformación de distancias o aproximaciones poligonales.
- Calibración de cámaras y reconstrucción 3D.
- Reconocimiento de objetos.
- Etiquetado de imágenes.
- Análisis de movimiento.
- Interfaz gráfica.

A continuación se detalla las principales características por las que está compuesta esta biblioteca, cómo son los módulos por los que está formada, los tipos de datos más importantes y las funciones más representativas.

2.9.1 Módulos

Las bibliotecas de OpenCV se corresponden con 4 módulos bien diferenciados:

- Módulo cv: Contiene las funciones principales.
- Módulo cvaux: Funciones auxiliares de OpenCV, también contiene funciones que se encuentran en grado de experimentación.
- Módulo cxcore: Contiene las estructuras de datos y el soporte para funciones de álgebra lineal.

- Módulo highgui: Contiene las funciones GUI.

Es importante no confundir las funciones, con los tipos de datos propios de OpenCV. Para ello, la propia biblioteca utiliza una sintaxis distinta para cada caso, con ligeras diferencias, aunque en principio si no se presta la debida atención, es fácil confundir ambas sintaxis.

Cada una de las funciones referenciadas en OpenCV comienza con las siglas “cv”, seguida del nombre de la función, con la primera letra de cada una de las palabras que componen dicho nombre en mayúscula. Por ejemplo: cvCreateImage, cvInvert, cvMatMulAdd, etc.

La sintaxis de los tipos de datos es muy similar a la de las funciones, aunque con la única diferencia de que los tipos comienzan con las siglas “Cv” y las funciones por “cv”. Por ejemplo: CvScalar, CvMat, etc. No obstante, existen algunos tipos que se declaran de forma totalmente distinta (IplImage). A continuación, se exponen los principales tipos de datos.

2.9.2 Tipos de datos en OpenCV

OpenCV proporciona tipos de datos básicos para su utilización. A continuación se describirán brevemente los más importantes:

- IplImage: Es el tipo de datos básico en OpenCV. Con este tipo de datos se representan todos los tipos de imágenes con sus componentes y características. Los campos de esta estructura ordenados son:

```
typedef struct  IplImage f  int nSiz e ; /* tamaño de la estructura iplImage */
int ID ; / *versión de la cabecera de la imagen */
int nChannels ;
int alphaChannel;
intdepth ; /* profundidad de la imagen en píxeles */
char colorMode l [ 4 ] ;
char channelSeq [ 4 ] ;
int dataOrder ;
int origin;
int align ; /* alineación de 4 u 8 bytes */
int width ;
int height ;
structIplROI _roi ; /* punter al a ROI si existe*_/
structIplImage_maskROI ; /* puntero a la máscara ROI ( si existe )*/
void_image Id ;  /*uso de la aplicación */
structIplTileInfo_tile Info ;
int imageSize ; /*tamaño útil en bytes */
char _ imageData ; /* puntero a la imagen alineada*/
int widthStep ; /* tamaño de alineamiento de linea en bytes*/
```

```
int BorderMode [ 4 ] ; /*modo de borde ( sup , inf , drch e izda )*/
int BorderCons t [ 4 ] ; /* constes . para borde sup , inf , drc h e izda .*/
char _ imageDataOrigin ; /* punter a la imagen sin alinear*/
IplImage ;
```

En la Tabla 2.9.1 se puede ver los componentes principales que forman esta estructura.

Es posible seleccionar algunas partes rectangulares de la imagen, lo que se conoce como regiones de interés (ROI). La estructura IplImage contiene el campo roi, que si no es nulo (NULL), apunta a la estructura IplROI, que contiene parámetros de la región seleccionada.

- CvArr: Es lo que se denomina un metatype, es decir, un tipo de dato ficticio que se utiliza de forma genérica a la hora de describir los parámetros de las funciones. CvArr* se utiliza para indicar que la función acepta arrays de más de un tipo.

- CvMat: Estructura empleada para operar con imágenes. Es una matriz que se caracteriza porque aparte de almacenar los elementos como cualquier matriz, ofrece la posibilidad de acceder a información adicional que puede resultar de gran utilidad.

```
typedef struct CvMatf int rows ; /* número de filas */
int cols ; /* número de columnas */
CvMatType type ; /*tipo de matriz */
int step; /*no se utiliza */
union ffloatfl ; /*puntero a los datos de tipo float */
double_db ; /*puntero a datos de doble precisión */
g data;
gCvMat
```

En todo programa implementado con OpenCV, este tipo de datos siempre irá asociado con la función cvCreateMat que permitirá configurar la estructura matricial de manera muy sencilla. Esta función se encarga de crear el encabezado de la imagen y de ubicar sus datos. Su estructura es:

```
CvMat_ cvCreateMa t ( int rows , int  cols , int type ) ;
rows : n úmero de filas de la matriz
cols : número de columnas de la matriz
type : tipo de lo s elementos de las matrices .
Se especifica de la forma :
CV <bit depth >(S jUj F )C<number of channels >.
Siendo :
bit depth : profundidad de bit ( 8 , 1 6 , 3 1 . . . )
number of channels : número de canales de matriz
( S jUj F ) : el tipo de datos de bit :
S : con signo
U: sin signo
F : flotante
```

- CvScalar: La estructura CvScalar es simplemente un vector de cuatro elementos. ´ Esta es muy útil a la hora de acceder a los píxeles de una imagen, sobre todo si es una imagen en color. La estructura CvScalar es la siguiente:

```
CvScalar double
v a l [ 4 ] ; / / v e c t o r 4D
```

- CvPoint: Define las coordenadas de un punto usando números enteros.

```
typedef struct CvPoint int x ; /*coordenada x */
int y ; /*coordenada y */
gCvPoint ;
```

CvPoint2D32f: Define las coordenadas de un punto usando punto flotante.

```
t ypedef struct CvPoint2D32f f  float x ; /* coordenada x */
float y ; /*c o o r d e n a d a y*/
gCvPoint2D32f ;
```

- CvSize: Estructura utilizada para definir las dimensiones de un rectángulo en píxeles.

```
typedef struct CvSizef  int width ; / *anchura del rectángulo(valor en píxeles*/
int height ; /* altura del rectángulo ( valor en píxeles ) */
g CvSize ;
```

OpenCV: IplImage	
Componente	**Descripción**
widhtStep	número de bytes entre puntos de la misma columna y filas sucesivas
nChannels	indica el número de canales de color de la imagen
*imageData	puntero a la primera columna de los datos de la imagen
widht,height	anchura y altura de la imagen en píxeles
depth	información sobre el tipo de valor de los píxeles Los posibles valores del campo depth son los siguientes: - IPL_DEPTH_8U: Enteros sin signo de 8 bits (unsigned char) - IPL_DEPTH_8S: Enteros con signo de 8 bits (signed char o char) - IPL_DEPTH_16S: Enteros de 16 bits con signo (short int) - IPL_DEPTH_32S: Enteros con signo de 32 bits (int) - IPL_DEPTH_32F: Punto flotante con precisión simple de 32 bits (float)

Tabla 2.9.1: Descripción de los componentes principales de IplImage.

Capítulo 3: Metodología de trabajo

En este capitulo se va a detallar brevemente la forma en la que se ha hecho uso de la teoría en la que se basa este proyecto, así como de las funciones que se han desarrollado y utilizado para conseguir los objetivos buscados.

3.1 Módulos del sistema

El desarrollo del proyecto se divide en distintos módulos (o pequeños programas) que realizan pequeñas tareas, que en conjunto, consiguen realizar la reconstrucción de la escena en 3D.

En la Figura 3.1.1 se puede observar un esquema con todos estos módulos colocados de manera secuencial.

3.2 Obtención de los parámetros de calibración

La calibración de la cámara se ha realizado de forma independiente al resto de las fases que prosiguen en el desarrollo del proyecto. Una vez realizada la calibración, ésta no será necesario volver a realizarla, siempre y cuando no se cambie la cámara con la que se adquieren las imágenes.

El primer paso antes de la ejecución del programa, consiste en tomar una serie de capturas, suficientemente grande, de un tablero de ajedrez tal y como se especificó en el capítulo anterior. Estas capturas han de realizarse de modo que el tablero vaya apareciendo con distintas orientaciones y posiciones a lo largo de la escena, siendo mejor cuanta mayor variabilidad de rotaciones y traslaciones existan del mismo.

De no tomar un número suficiente de imágenes, los valores que se obtendrían de los parámetros intrínsecos y de distorsión, podrían ser poco precisos, pudiéndose observar en la Figura 3.2.1 un ejemplo de lo que se obtendría al rectificar la imagen con los parámetros de la cámara.

Lo más importante a la hora de realizar estas capturas, es que se puedan apreciar todas las esquinas interiores del tablero, tal y como se muestra en la Figura 3.2.2.

A continuación se muestran algunos fragmentos con las funciones más significativas del código referente al módulo de calibración.

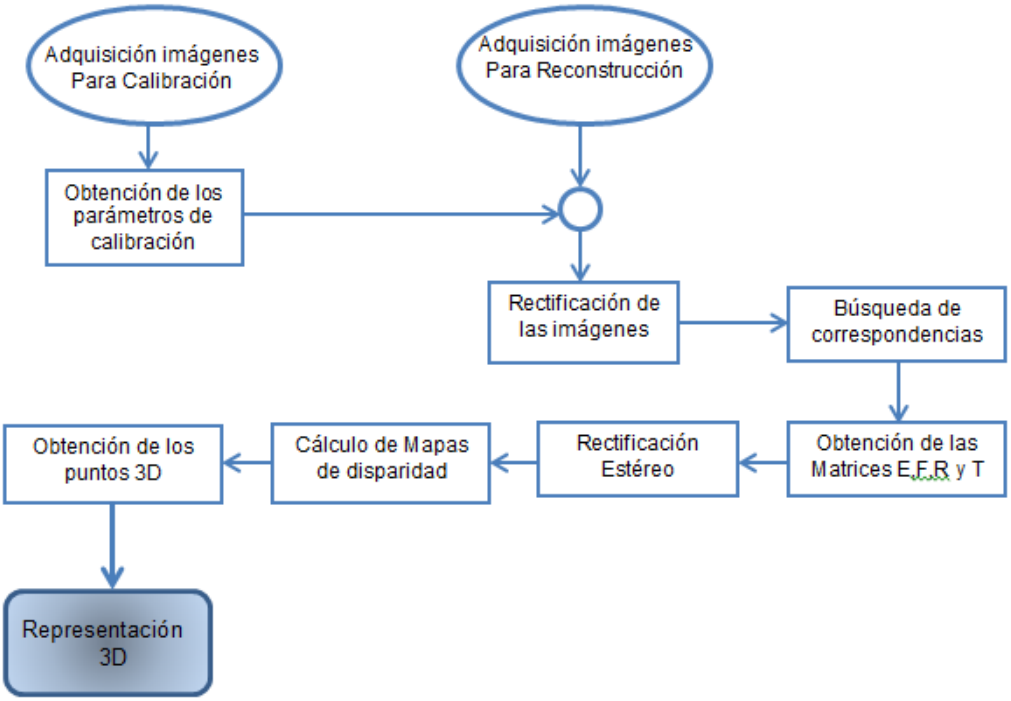


Figura 3.1.1: Esquema con el proceso de trabajo realizado de forma secuencial.

El programa llama a esta función para cada imagen del tablero capturada

```
found = cvFindChessboardCorners(gray_image, board_sz, corners,
&corner_count,CV_CALIB_CB_ADAPTIVE_THRESH|CV_CALIB_CB_NORMALIZE_IMAGE)
```

la cual trata de encontrar las coordenadas de las esquinas interiores del tablero, devolviendo el valor de 1, si se han encontrado todas.

Las variables que forman parte de esta función son:

- gray_image: Puntero a una matriz que contiene la imagen con el tablero del que se quieren encontrar las esquinas interiores.
- board_sz: Indica el tamaño que tiene el tablero(número de esquinas en cada dirección, x e y).
- corners: es la dirección de una matriz que contiene las coordenadas de las esquinas detectadas.
- &corner_count: dirección que indica el número de esquinas detectadas.
- `CV_CALIB_CB_ADAPTIVE_THRESH|CV_CALIB_CB_NORMALIZE_IMAGE`: indica el umbral de intensidad.

Figura 3.2.1: Ejemplo de rectificación cuando no se tienen suficientes imágenes para realizar la calibración (imagen izquierda) y cuando los parámetros de la cámara están bien estimados (imagen derecha).

A continuación, se ejecuta la siguiente función que lo que busca es obtener una mejora en la detección de las coordenadas de las esquinas:

```
cvFindCornerSubPix(gray_image,corners,corner_count,cvSize(11,11),cvSiz
e(-1,-1), cvTermCriteria(CV_TERMCRIT_EPS+CV_TERMCRIT_ITER,30,0.1 ));
```

que básicamente tiene las mismas variables que la función anterior, con la salvedad de que en este caso se realiza una búsqueda dentro de un rectángulo de tamaño cvSize(ancho,alto), y con un criterio determinado por la variable cvTermCriteria.

Para comprobar si las esquinas encontradas por cvFindChessboardCorners() son correctas se llama a la siguiente función:

```
cvDrawChessboardCorners(image, board_sz, corners,corner_count, found);
```

que lo que realiza es dibujar sobre las imágenes de los tableros las esquinas detectadas. Siendo de gran ayuda para el usuario el comprobar si los valores detectados coinciden con los valores reales.

Las variables que son necesarias en esta función son semejantes a la función anterior, con la especificación de la última, ya que si found=1, se dibujarán las esquinas, según se indica en la Figura 3.2.3 (a) y en caso de ser found=0, se dibujarían según indica la Figura 3.2.3 (b).

En el caso de que un found=0, fuese devuelto, el programa se interrumpiría, siendo necesario eliminar esta imagen del proceso de calibración.

Una vez detectadas las esquinas de los tableros, se ejecuta la función propiamente dicha que realiza la calibración y nos permite obtener los parámetros de la cámara.

```
cvCalibrateCamera2(object_points,image_points,point_counts,foto,
intrinsic_matrix,distortion_coeffs,NULL, NULL,0);
```

en la que los 3 primeros parámetros son matrices que tienen la información de la posición de los puntos obtenidos de los tableros, el número de puntos por imagen y el número total de puntos encontrados. El siguiente es el tamaño de la imagen y los dos siguientes son las matrices en las que se almacenan los parámetros intrínsecos y de distorsión de la cámara.

Finalmente, de este proceso se obtienen dos ficheros, Intrinsics.xml y Distortion.xml, que contienen la información de los parámetros de la cámara, que posteriormente vamos a necesitar para realizar transformaciones a las imágenes y para obtener otros parámetros y datos de interés.

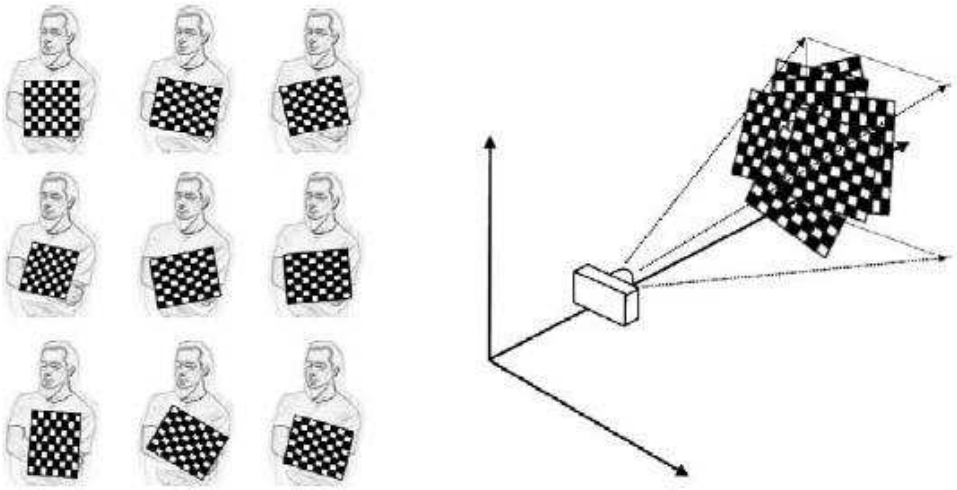
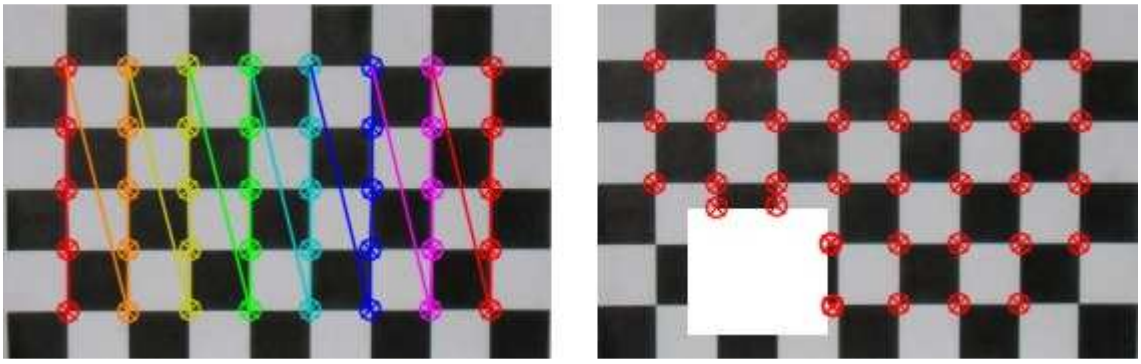


Figura 3.2.2: Esquema de captura de imágenes, variando la posición del tablero a lo largo de la escena.



(a) (b)

Figura 3.2.3: Ejemplo de tableros en el proceso de calibración. En el caso (a), el tablero es correctamente detectado, mientras que como se aprecia en el caso (b), no se detectan todas las esquinas.

3.3 Búsqueda de correspondencias

El proceso de obtención de las correspondencias entre imágenes, como se ha podido observar en la teoría es un proceso complejo y secuencial que consta de gran parte de elementos diferenciados.

Para el desarrollo de la solución de este proyecto, se ha partido de una solución existente para dicha tarea, incluida en los ejemplos de OpenCV, que ha sido propiamente modificada en función de nuestras necesidades.

Las principales funciones que se utilizan en este módulo para calcular las correspondencias entre las dos imágenes son:

```
cvExtractSURF( im_left, 0, &LeftImageKeypoints, &LeftImageDescriptors,
storage, params );
```

En la que los parámetros de entrada son las imágenes (puesto que esta función se llama para cada imagen) y como parámetros de salida son los puntos de interés de cada una, así como los descriptores.

```
findPairs( LeftImageKeypoints, LeftImageDescriptors,
RightImageKeypoints, RightImageDescriptors, ptpairs );
```

La cual se encarga de buscar las parejas de puntos homólogos comparando ambas imágenes (en concreto sus keypoints y sus descriptores).

Este proceso se tiene que repetir para cada par de imágenes de los que queramos obtener los puntos de interés.

El algoritmo tiene más parámetros y más funciones, pero estas son las que hacen las tareas básicas de obtención del algoritmo. Internamente cada función está compuesta por llamadas a otras muchas funciones de OpenCV para calcular estos parámetros.

Observando los códigos de las funciones se puede observar de lo que se habla en el párrafo anterior, pero que no tiene transcendencia a la hora de explicar el desarrollo del método de cálculo de correspondencias.

3.4 Obtención de las matrices F y E

La obtención de las matrices fundamental y esencial, con la consiguiente información que ellas acarrean, se puede obtener ahora que ya tenemos una correspondencia con puntos homólogos tras haber ejecutado el módulo de SURF.

Este módulo es relativamente sencillo, y ya disponemos prácticamente de toda la información necesaria, ya que para obtener la matriz fundamental, solo necesitamos como información previa los puntos de interés calculados previamente.

Con toda la información disponible y con el acondicionamiento para su uso, podemos ejecutar la siguiente función proporcionada por la biblioteca de las OpenCV.

```
cvFindFundamentalMat(int_point1,int_point2,Fundamental,CV_FM_RANSAC,
1.0,0.99,NULL);
```

cuyos parámetros de entrada son los puntos de interés calculados en ambas imágenes y asociados 1 a 1 entre ellos (int_point1 e int_point2), y lo demás son una matriz en la que se guardarán los parámetros de la matriz fundamental y el método que se seguirá para calcularlo, las diferentes opciones se pueden observar en la Tabla 3.5.1. Como elemento de salida, esta función nos devuelve un 1, si la matriz fundamental ha podido calcularse, ó 0 en el caso de que no se haya podido calcular.

La matriz esencial E, por el contrario, se calcula simplemente mediante la aplicación del álgebra y de relaciones matriciales, por lo que la programación de este módulo no dispone de ninguna función especial de librería, simplemente se ha procedido a realizar el desarrollo de la teoría matemática.

La matriz esencial se obtiene, a partir de la matriz fundamental, a partir de la siguiente ecuación:

$$E = K^t F K \qquad (3.1)$$

siendo F la matriz fundamental y K la matriz de parámetros intrínsecos de la cámara.

Ahora, partiendo del conocimiento de que $E = RT$, se procede a realizar la descomposición de valores singulares de la matriz E, para así poder obtener las matrices R y T (rotación y traslación) que relacionan la posición relativa existente entre ambas cámaras.

Nombre del método	Número de puntos	Algoritmo
CV_FM_7POINT	N=7	7-point
CV_FM_8POINT	N≥8	8-point
CV_FM_RANSAC	N≥8	Ransac
CV_FM_LMEDS	N≥8	LMedS

Tabla 3.4.1: Métodos de obtención de la matriz fundamental

El proceso comienza por obtener los valores de la DVS, esto es, las matrices U, Σ y V, con lo que obtendremos las matrices como sigue:

$$T = U \cdot \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot U^t \qquad (3.2)$$

$$R = U \cdot \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot V^t \qquad (3.3)$$

Al ser un proceso iterativo, el resultado obtenido no se asemeja con el resultado óptimo real. Por ello, la matriz de traslación T, no tiene porque ser una matriz con 0 en los elementos diagonal, por lo que en el programa realizado, se fuerza a que dichos valores sean 0.

3.5 Rectificación estéreo de las imágenes

Para realizar el proceso de rectificación estéreo, necesitamos una serie de parámetros calculados previamente, como son: El conjunto de puntos de correspondencias obtenidos mediante el módulo SURF, así como los parámetros de la cámara, ya que se va a proceder a rectificar las imágenes a la par que se busca correspondencia unidimensional de los puntos de interés obtenidos.

Para ello lo que realizamos es llamar a la función

```
cvStereoRectifyUncalibrated(interes1,interes2,Fundamental,tamano,H1,H2
,5);
```

la cual tiene como parámetros de entrada las coordenadas de los puntos de interés obtenidos con SURF, así como la matriz fundamental, y lo que obtenemos son H1 y H2, que son las matrices de homografía que permite recalcular la posición de las imágenes con ayuda de las siguientes funciones.

Mediante las siguientes multiplicaciones de matrices, eliminamos los problemas debidos a la cámara

$$R_1 = M^{-1}H_1M$$

$$R_2 = M^{-1}H_2M$$

y mediante la función `cvInitUndistortRectifyMap(M1,D1,R1,M1,mx1,my1);` eliminamos los errores debidos a la distorsión de la cámara.

En mx1 y my1 se introducen las coordenadas de mapeo para los nuevos puntos de la imagen y con ayuda de la función

```
cvRemap( img1, imagen_rectificada1, mx1, my1 );
```

finalmente se obtienen las imágenes rectificadas y libres de distorsiones.

Como se puede observar en la figura 3.5.1 se obtiene un par estéreo paralelo cuya diferencia de valores solo difieren entre sí en sus coordenadas x.

Figura 3.5.1: Par estéreo rectificado mediante el método de Hartley.

3.6 Mapas de disparidad

Calculando el mapa de disparidad podemos obtener una visión más global del sistema que tratamos de reconstruir, ya que nos permite obtener la profundidad a la que se encuentran los objetos, y con ello, la coordenada Z en la que se sitúa cada punto estudiado.

Para calcular el mapa de disparidad de cada par de imágenes, necesitamos tener un par estéreo correctamente rectificado y situado en posición paralela (algoritmo de Hartley).

Una vez disponemos de todos los ingredientes para hallar el mapa de disparidades, llamamos a la función

```
cvFindStereoCorrespondenceBM(  imagen_rectificada1,imagen_rectificada2,
disp,BMState);
```

En la que se introducen como entrada las imágenes rectificadas de forma estéreo, y un estado especial, BMState, el cual tiene muchos parámetros que permiten el cálculo del mapa de profundidades. Y como salida obtenemos disp, que es el mapa de disparidad.

La variable BMState está compuesta por una serie de variables que son las que detallan la forma en la que se calcula el mapa de disparidad.

A continuación se explican un poco cada variable en qué influye:

- BMState->preFilterSize

- BMState->preFilterCap

Estas dos primeras variables son filtros que se aplican a la imagen original con el fin de normalizer iluminaciones y contrastes entre ambas imágenes (en este proyecto, ya que se toman las imágenes bajo las mismas condiciones, no resultan de mucho interés)

- BMState->SADWindowSize

La variable SADWindowSize nos indica el tamaño de la ventana sobre la que se va a aplicar el algoritmo de correlación de suma de diferencias absolutas.

Esta variable tiene gran importancia, y depende del grado de texturas que existan en la imagen el que tome valores elevados o bajos, ya que lo que realiza es una discriminación de pequeñas texturas que pueden conllevar a errores debido a una no correcta rectificación.

Para un caso concreto como el de la figura 3.6.1 se puede observar el efecto de aumentar el tamaño de la ventana SAD, eliminándose así pequeñas diferencias que “ensucian” el mapa de disparidades.

- BMState->minDisparity

Esta variable lo que nos permite es modificar la diferencia de profundidad que puede existir entre la imagen tomada con la cámara izquierda y con la cámara derecha. En nuestro caso de estudio se intenta que la captura de las imágenes se haga lo más paralelamente posible con el fin de minimizar errores y maximizar zona común de correspondencias, por lo que se toma un valor de 0 para esta variable.

- BMState->numberOfDisparities

Esta variable nos indica el número de píxeles en los que tenemos que buscar correspondencias. Cuanto mayor sea este valor, menos preciso será el resultado. Un valor elevado de esta variable se vuelve necesario cuando la rectificación es mala.

A pesar de que existen otras muchas variables, para el caso de estudio no resultan interesantes, por lo que no se procederá a la explicación de las mismas.

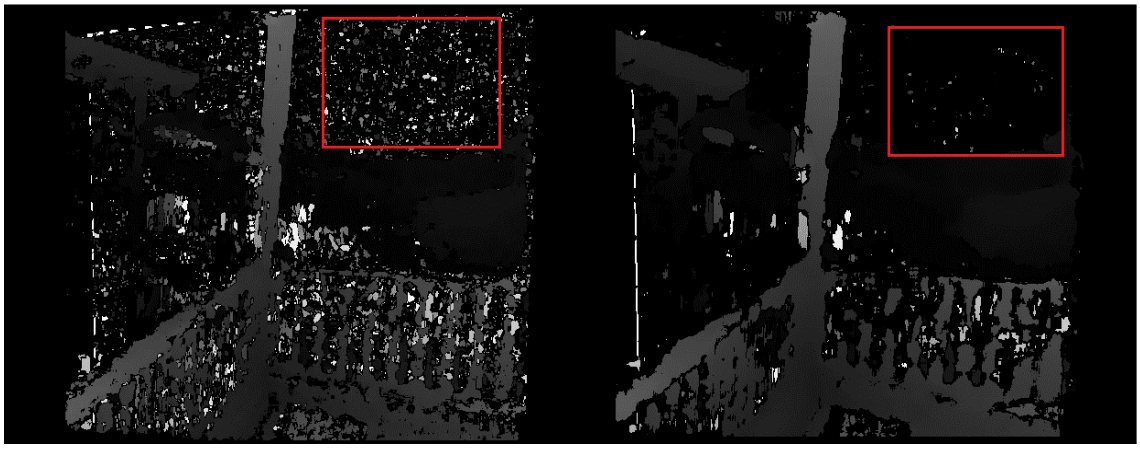


Figura 3.6.1: Variación del parámetro SADWindowSize (valor de 9 en la imagen izquierda y de 17 en el de la derecha). Se puede apreciar como las pequeñas diferencias se eliminan a costa de perder precisión.

3.7 Obtención de las coordenadas 3D

Una vez obtenidos los mapas de disparidad, la siguiente tarea lógica consiste en obtener las coordenadas tridimensionales de los puntos en el espacio real, es decir, la reproyección de dichos puntos en el plano real a partir de sus proyecciones en el pano imagen.

Esta tarea la realiza una función de OpenCV llamada

```
cvReprojectImageTo3D(mapa_disparidad,recons3D, Q);
```

en la que las variables de entrada son:

- mapa_disparidad: mapa de profundidades del par estéreo.
- recons3D: matriz en la que se guardarán las coordenadas (x,y,z) de los puntos obtenidos.

- Q: matriz de reproyección, que se encarga de hacer las transformaciones oportunas entre el plano imagen y el plano físico.
  Está compuesta por parámetros que relacionan el par estéreo de las imágenes, como por ejemplo la traslación de la imagen derecha sobre la izquierda o las distancias focales.

  Esta función procesa la siguiente información:

$$p_x = x - C_x \tag{3.1}$$

$$p_y = y - C_y \tag{3.2}$$

$$p_z = -1/T_x \tag{3.3}$$

$$p_w = d \cdot f + \frac{C_x - C_x'}{T_x} \tag{3.4}$$

y estas funciones permiten obtener las coordenadas del mundo real mediante el cociente de:

$$P_x = p_x/p_w \tag{3.5}$$

$$P_y = p_y/p_w \tag{3.6}$$

$$P_z = p_z/p_w \tag{3.7}$$

Donde P_i son las coordenadas reales del mundo.

3.8 Representación de los resultados

Una vez que ya hemos obtenido las coordenadas de los puntos, lo que necesitamos es un interfaz en el que representarlos. Para ello, vamos a proceder a utilizar un visualizador contenido en las librerías Point Cloud Library, que son un proyecto similar al de OpenCV en el que se consiguen optimizar los tiempos de cómputo y se centra mucho más que OpenCV en la representación 3D de los puntos.

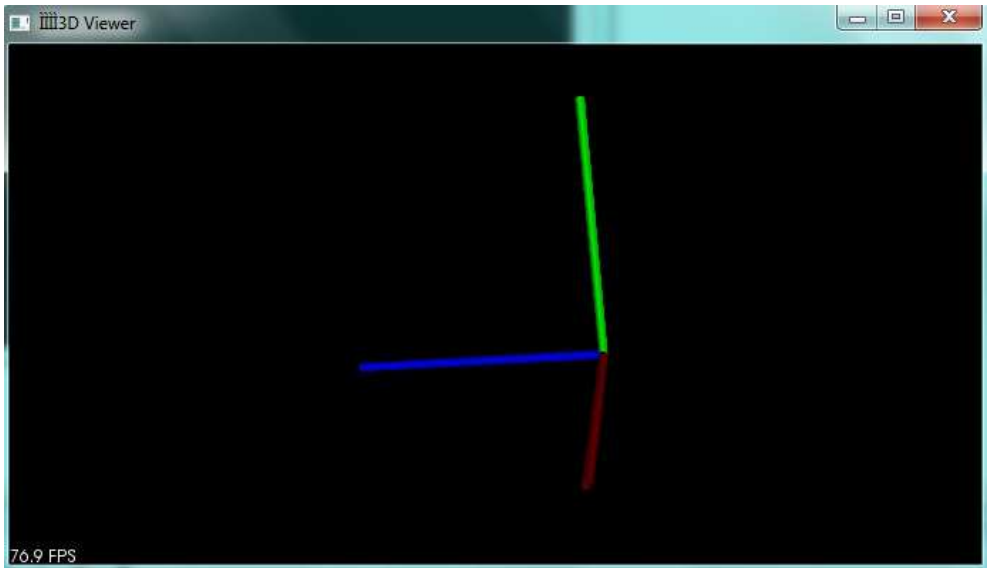


Figura 3.8.1: Interfaz gráfica que permite realizar la representación tridimensional de los puntos calculados

Para ello utilizamos las siguientes sentencias que se encargan de crear un visualizador como el que se muestra en la figura 3.8.1.

```
boost::shared_ptr<pcl::visualization::PCLVisualizer> createVisualizer
(pcl::PointCloud<pcl::PointXYZRGB>::ConstPtr cloud)
{
  boost::shared_ptr<pcl::visualization::PCLVisualizer> viewer (new
pcl::visualization::PCLVisualizer ("3D Viewer"));
  viewer->setBackgroundColor (0, 0, 0);
  pcl::visualization::PointCloudColorHandlerRGBField<pcl::PointXYZRGB>
rgb(cloud);
  viewer->addPointCloud<pcl::PointXYZRGB> (cloud, rgb,
"reconstruction");
 viewer->setPointCloudRenderingProperties
(pcl::visualization::PCL_VISUALIZER_POINT_SIZE, 3, "reconstruction");
  viewer->addCoordinateSystem ( 1.0 );
  viewer->initCameraParameters ();
  return (viewer);
      }
```

Básicamente lo que realiza es crear un sistema de ejes coordenados en los que se representarán posteriormente los puntos que hemos calculado previamente.

Capítulo 4: Resultados

En este capítulo vamos a poner de manifiesto una serie de pruebas y resultados obtenidos durante todo el proyecto, permitiendo obtener datos y resultados que nos ayuden a obtener las conclusiones pertinentes sobre el algoritmo y sus capacidades.

Con el fin de conocer qué módulos son los que incurren en mayor consumo de tiempo, se incluirá un estudio del tiempo que tarda en ejecutarse cada módulo por separado, pudiéndose así conocer de manera cuantitativa que módulos podrían ser motivo de estudio para futuras mejoras y optimizaciones del proyecto.

El procesado de todo el sistema se ha realizado con un ordenador con procesador Intel i7 de 2,20 GHz. Esta característica se vuelve importante de cara a los tiempos de cómputo de los módulos.

4.1 Resultados de la calibración

El módulo inicial previo a la realización de la reconstrucción es el de la calibración de la cámara.

Para determinar el número mínimo de imágenes que se necesitan para la calibración se muestran unos gráficos en los que se aprecia que para valores superiores de 16 imágenes los resultados son suficientemente buenos y las variaciones de los valores son mínimas, por lo que se puede concluir que a partir de la toma de series de 16 imágenes los resultados obtenidos son aceptables para continuar con el proceso de calibración.

Este dato resulta interesante, puesto que la teoría nos indicaba que con poco más de 2 imágenes se podían obtener los parámetros de forma adecuada.

Los resultados que obtenemos de este módulo, como se ha explicado anteriormente, son las matrices de parámetros intrínsecos K y de distorsión D.

A continuación se presentan los resultados obtenidos para diferentes cámaras y a diferentes resoluciones. Esto nos permite hacernos una idea de los tiempos de cómputo necesarios para distintos tamaños de imagen.

Figura 4.1.1: Conjunto 1 de vistas para la calibración

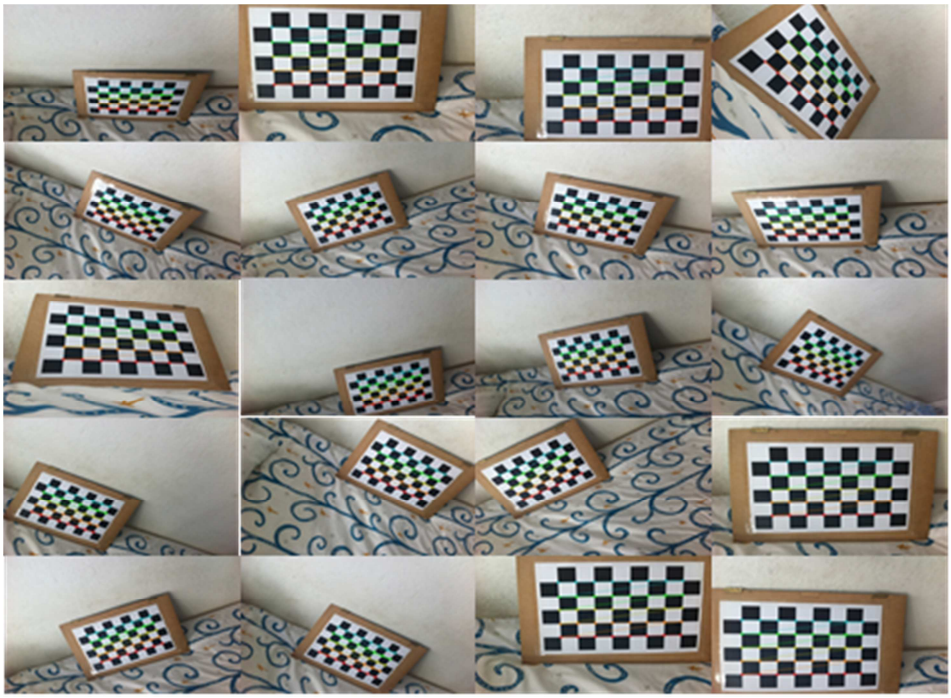


Figura 4.1.2: Conjunto 1 de vistas obtenidas durante el proceso de calibración, en ellas se puede observar como se ha conseguido encontrar las esquinas del tablero en todas las vistas.

Figura 4.1.3: Conjunto 2 de imágenes para la calibración

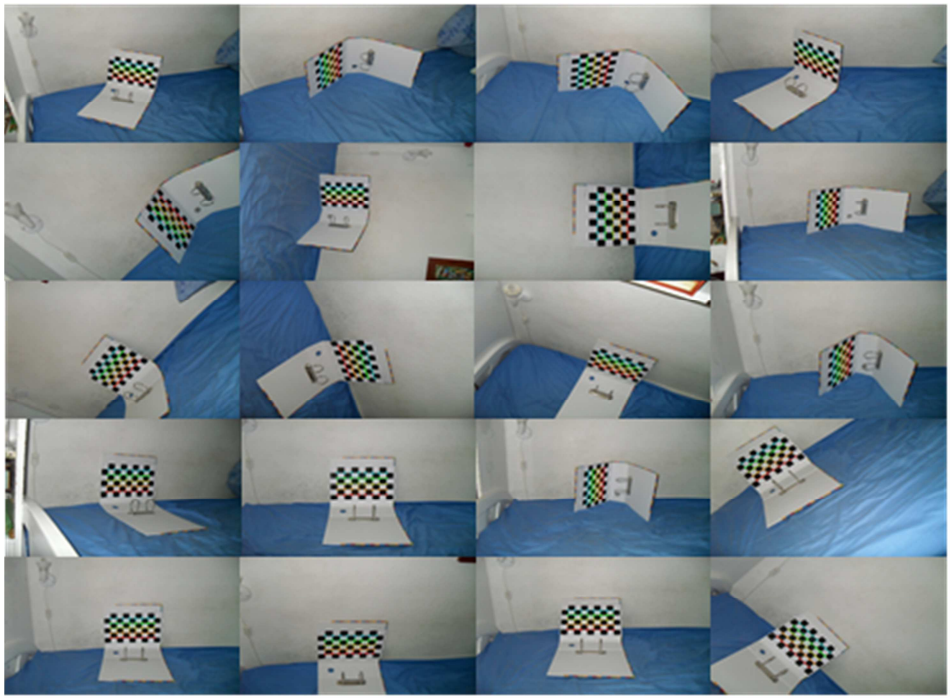


Figura 4.1.4: Conjunto 2 de imágenes tras el proceso de calibración. En ella se puede observar la búsqueda de esquinas realizada de manera satisfactoria

Conjunto 1 → Cámara 1, Resolución 800x598, Conjunto de datos 1:

En la Figura 4.1.1 y Figura 4.1.2 se presentan el conjunto de imágenes sin calibrar y tras pasar por el programa de calibrado.

Los resultados obtenidos de la calibración de este conjunto de datos son:

$$K = \begin{pmatrix} 752{,}68 & 0 & 390{,}6 \\ 0 & 755{,}2 & 293{,}89 \\ 0 & 0 & 1 \end{pmatrix}$$

$$D = (0{,}043 \quad 0.189 \quad -0{,}007 \quad -0{,}0022 \quad 1.74)$$

Los tiempos de cómputo para esta resolución y para distinto número de imágenes son:

Nº Imágenes	**2**	**4**	**6**	**8**	**10**	**12**	**14**	**16**	**18**	**20**
Tiempo (s)	0,16	0,48	0,74	0,9	1,27	1,49	1,71	1,99	2,41	2,49

Tabla 4.1.1: Tiempos de cómputo para el módulo de calibración con imágenes de 800x598

Unos resultados interesantes son los gráficos de distorsión correspondientes a la Figura 4.1.5 y Figura 4.1.6, que nos muestran la distribución de la distorsión a medida que aumenta la distancia.

Y por último en la Figura 4.1.7 y Figura 4.1.8 se presentan los gráficos de los que se concluye que 15 imágenes son suficientes para considerar unos buenos parámetros de calibración para esta resolución.

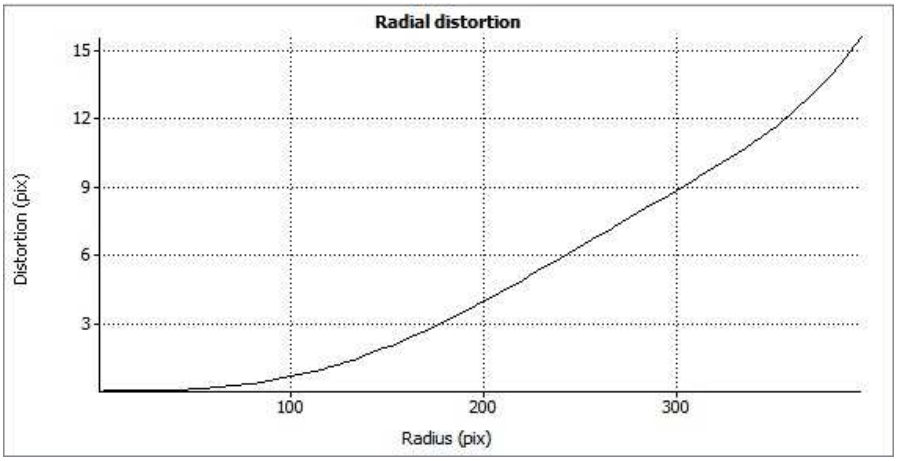


Figura 4.1.5: Gráfico de la distorsión Radial para imagen de 800x598

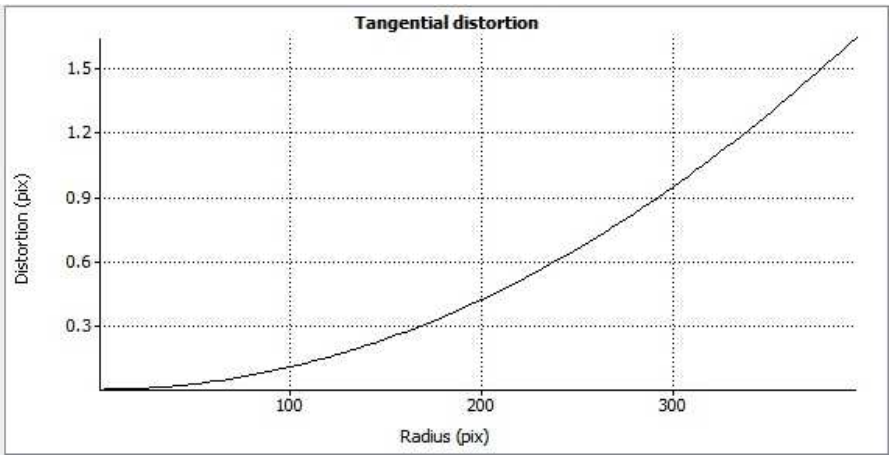


Figura 4.1.6: Gráfico de la distorsión tangencial para imagen de 800x598

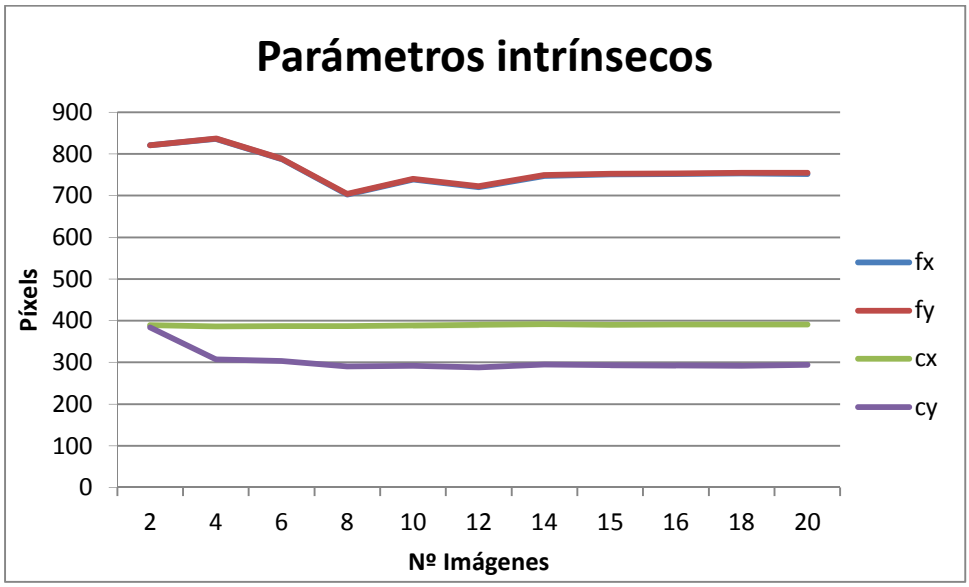


Figura 4.1.7: Gráfico de Parámetros intrínsecos en función del número de imágenes para 800x598

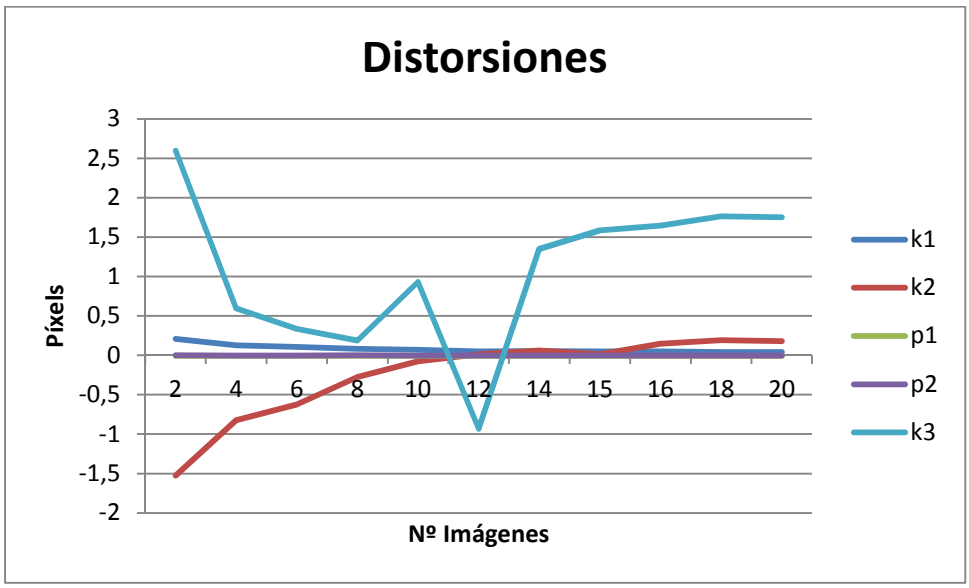


Figura 4.1.8: Gráfico de Distorsiones en función del número de imágenes para 800x598

Conjunto 2 → Cámara 1, Resolución 2592x1936, Conjunto de datos 1:

En la Figura 4.1.1 y Figura 4.1.2 se presentan el conjunto de imágenes sin calibrar y tras pasar por el programa de calibrado al igual que en el caso anterior, ya que se trata de las mismas imágenes pero con diferentes resoluciones.

Los resultados obtenidos de la calibración de este conjunto de datos son:

$$K = \begin{pmatrix} 1170{,}4 & 0 & 1295{,}5 \\ 0 & 1137{,}4 & 967{,}45 \\ 0 & 0 & 1 \end{pmatrix}$$

$$D = (0{,}00042 \quad 0{,}000001 \quad 0{,}0005 \quad 0{,}0009 \quad 0)$$

Los tiempos de cómputo para esta resolución y para distinto número de imágenes son:

Nº Imágenes	**2**	**4**	**6**	**8**	**10**	**12**	**14**	**16**	**18**	**20**
Tiempo (s)	8,4	36,5	104,2	121,3	204,1	242,1	256,1	359,7	403,9	428,1

Tabla 4.1.2: Tiempos de cómputo para el módulo de calibración con imágenes de 2592x1936

Unos resultados interesantes son los gráficos de distorsión correspondientes a la Figura 4.1.9 y Figura 4.1.10, que nos muestran la distribución de la distorsión a medida que aumenta la distancia.

Y por último en la Figura 4.1.11 y Figura 4.1.12 se presentan los gráficos de los que se concluye que 15 imágenes son suficientes para considerar unos buenos parámetros de calibración.

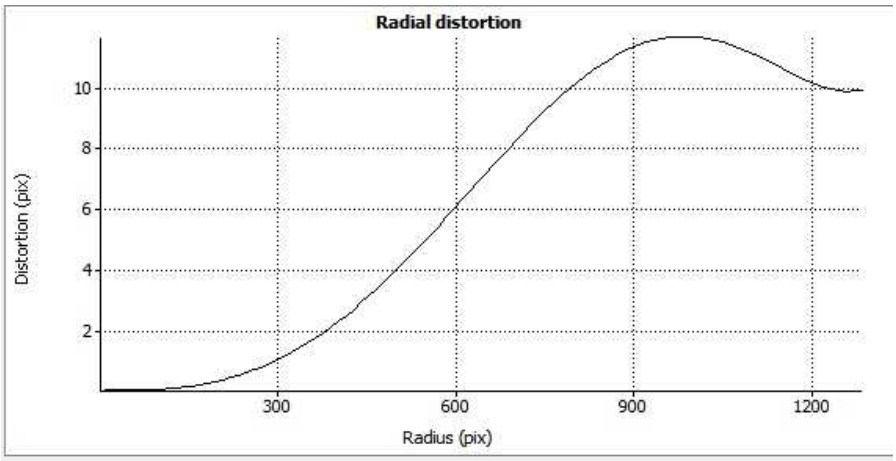


Figura 4.1.9: Gráfico de la distorsión Radial para imagen de 2592x1936

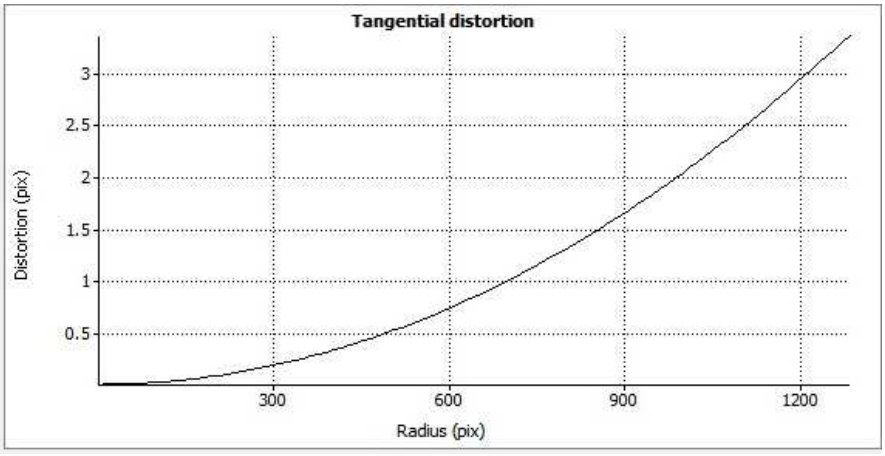


Figura 4.1.10: Gráfico de la distorsión Tangencial para imagen de 2592x1936

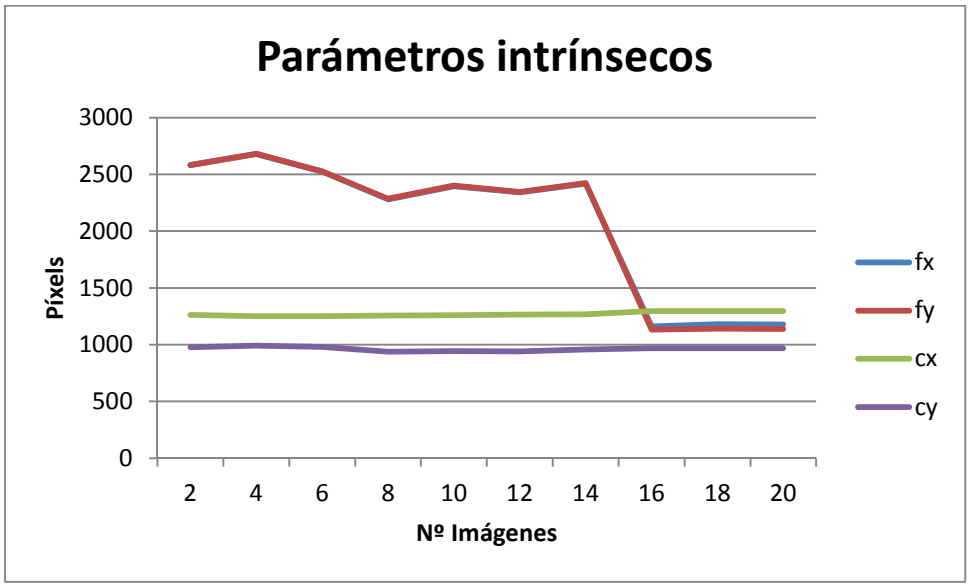


Figura 4.1.11: Gráfico de Parámetros intrínsecos en función del número de imágenes para 2592x1936

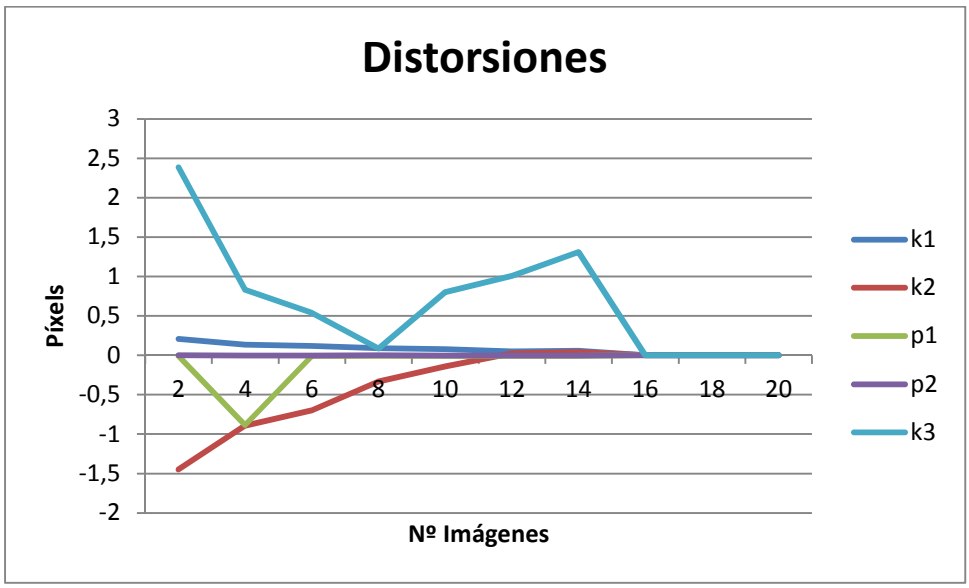


Figura 4.1.12: Gráfico de Distorsiones en función del número de imágenes para 2592x1936

Conjunto 3 → Cámara 2, Resolución 1024x768 , Conjunto de datos 2:

En la Figura 4.1.3 y Figura 4.1.4 se presentan el conjunto de imágenes sin calibrar y tras pasar por el programa de calibrado para este caso de estudio.

Los resultados obtenidos de la calibración de este conjunto de datos son:

$$K = \begin{pmatrix} 1109{,}97 & 0 & 509{,}82 \\ 0 & 1114{,}24 & 374{,}57 \\ 0 & 0 & 1 \end{pmatrix}$$

$$D = (-0{,}264 \quad 1{,}935 \quad 0{,}003 \quad -0{,}006 \quad 8{,}697)$$

Los tiempos de cómputo para esta resolución y para distinto número de imágenes son:

Nº Imágenes	**2**	**4**	**6**	**8**	**10**	**12**	**14**	**16**	**18**	**20**
Tiempo (s)	0,81	1,41	2,07	3,66	4,41	5,04	5,84	7,51	8,65	9,74

Tabla 4.1.3: Tiempos de cómputo para el módulo de calibración con imágenes de 1024x768

Unos resultados interesantes son los gráficos de distorsión correspondientes a la Figura 4.1.13 y Figura 4.1.14, que nos muestran la distribución de la distorsión a medida que aumenta la distancia.

Y por último en la Figura 4.1.15 y Figura 4.1.16 se presentan los gráficos de los que se concluye que 15 imágenes son suficientes para considerar unos buenos parámetros de calibración.

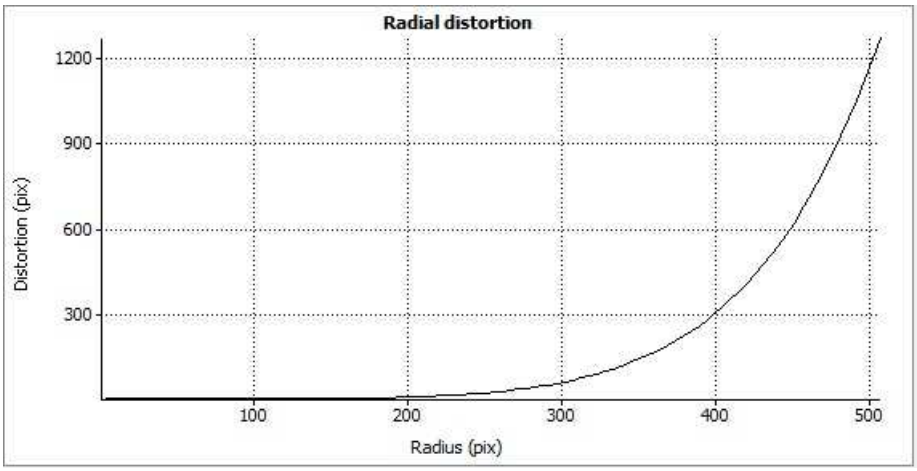


Figura 4.1.13: Gráfico de la distorsión Radial para imagen de 1024x768

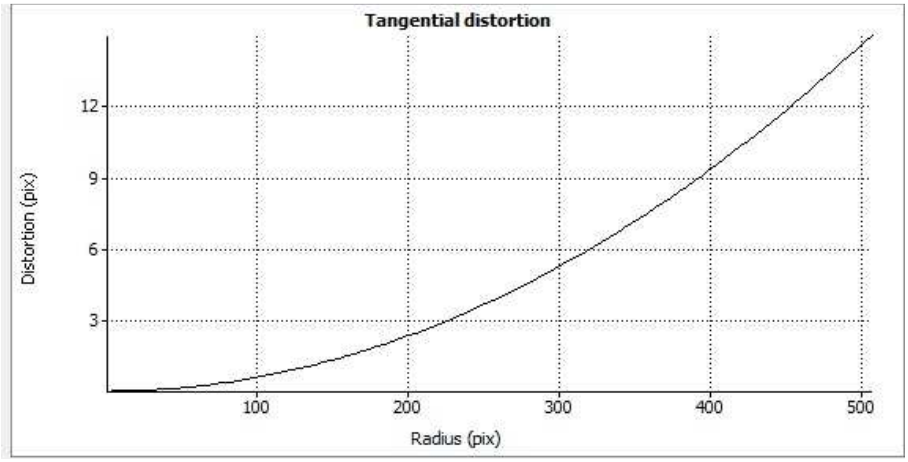


Figura 4.1.14: Gráfico de la distorsión Tangencial para imagen de 1024x768

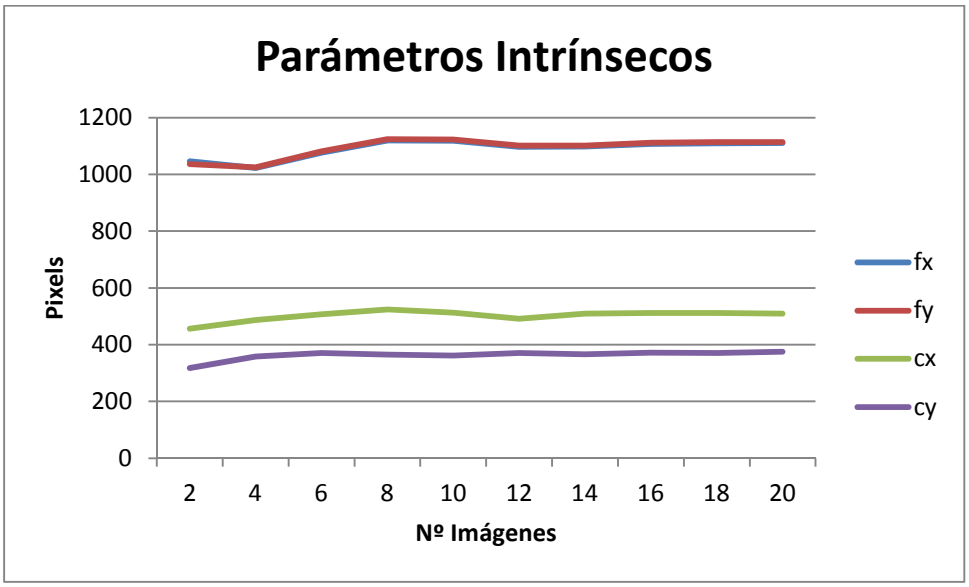


Figura 4.1.15: Gráfico de los parámetros intrínsecos de la cámara en función del número de capturas realizadas para una resolución de 1024x768

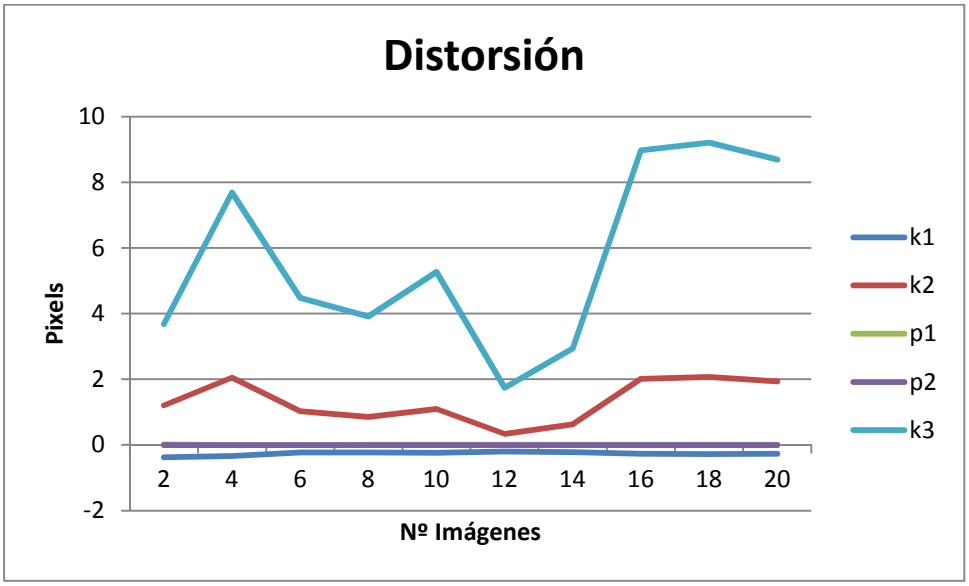


Figura 4.1.16: Gráfico de los parámetros intrínsecos de la cámara en función del número de capturas realizadas para resolución de 1024x768

Figura 4.2.1: Imagen antes de la rectificación (izquierda) y después de la rectificación (derecha)

Como puede observarse, los tiempos necesarios para el cálculo de los parámetros de calibración, no son muy elevados para imágenes con resoluciones bajas (800x598), pero para resoluciones elevadas, se vuelve un inconveniente el tiempo, por lo que probablemente se descartarían imágenes con estas resoluciones para aplicaciones en las que el tiempo juegue un papel importante.

4.2 Resultados de la Rectificación

El paso siguiente tras la calibración es la rectificación de las imágenes. Este es el primer proceso que se lleva a cabo en el módulo de Reconstrucción.

La rectificación de las imágenes se computa con la ayuda de los parámetros obtenidos de la calibración.

Debido a que los parámetros de la cámara son bastante aceptables, la rectificación de las imágenes no afecta visiblemente a los resultados antes y después de aplicar la rectificación como se puede apreciar en la Figura 4.2.1.

El tiempo de cómputo de este módulo es bastante bajo en comparación con los demás, ya que se trata de un módulo relativamente sencillo y con pocas operaciones. En la Tabla 4.2.1 se presenta una comparativa de los tiempos utilizados en la rectificación individual de cada imagen de los diferentes Sets.

Imágenes	Set1	Set2	Set3	Set4	Set5	Set6
Tiempo (ms)	34	314	44	316	38	34

Tabla 4.2.1: Tiempos de ejecución del proceso de rectificación

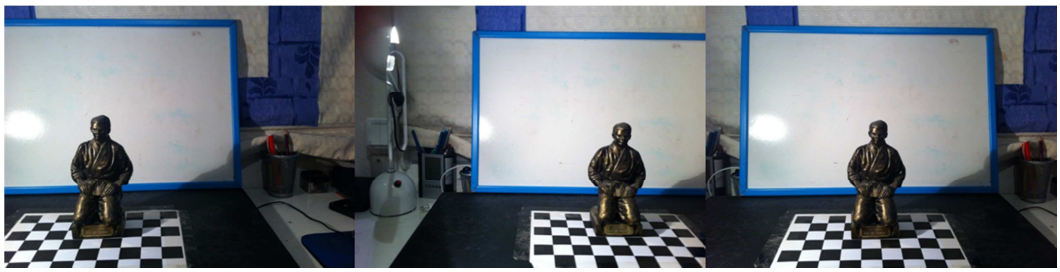


Figura 4.2.2: Conjunto de imágenes que forman el Set1 (800x598) y Set 2 (2592x1936)

Figura 4.2.3: Conjunto de vistas correspondientes al Set 3 (800x598) y al Set 4 (2592x1936)

Figura 4.2.4: Conjunto de vista relativas al Set 5 (800x598)

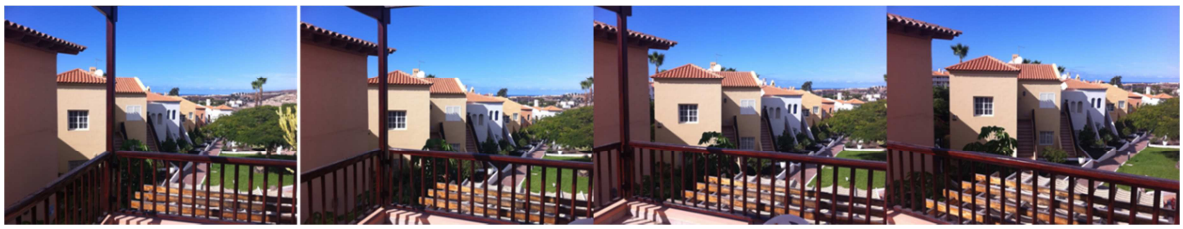


Figura 4.2.5: Conjunto de vistas correspondientes con el Set 6 (800x598)

En las Figuras 4.2.2, 4.2.3, 4.2.4 y 4.2.5 se presentan los conjuntos de imágenes utilizados para la realización de pruebas y obtención de resultados de este proyecto, y a los que se harán referencia en apartados posteriores.

4.3 Resultados de Búsqueda de correspondencias

El proceso de búsqueda de correspondencias cobra gran importancia debido a que es el paso clave que permite obtener los parámetros que relacionan el par estéreo. Sin un buen resultado en este proceso, la reconstrucción se vuelve prácticamente imposible, ya que los cálculos posteriores no arrojarían resultados coherentes de relaciones estereoscópicas.

El parámetro clave que permite obtener unos buenos resultados se trata del valor de la matriz Hessiana, el cual limita la aceptación de los puntos de interés a todos aquellos valores que superan el valor predefinido por el usuario. En los textos se recomienda que este valor se encuentre entre el rango 300-500, y como se observará a continuación se cumple muy bien este criterio.

En la Figura 4.3.1 se presentan la relación entre la imagen con los puntos de interés tomados como válidos y a su vez, una representación del mapa de disparidad que finalmente se obtendría si se utilizase ese valor.

Posteriormente, en las figuras siguientes de este apartado 4.3, se presentan diversos ejemplos con el número de puntos de interés obtenidos en función del parámetro dicho.

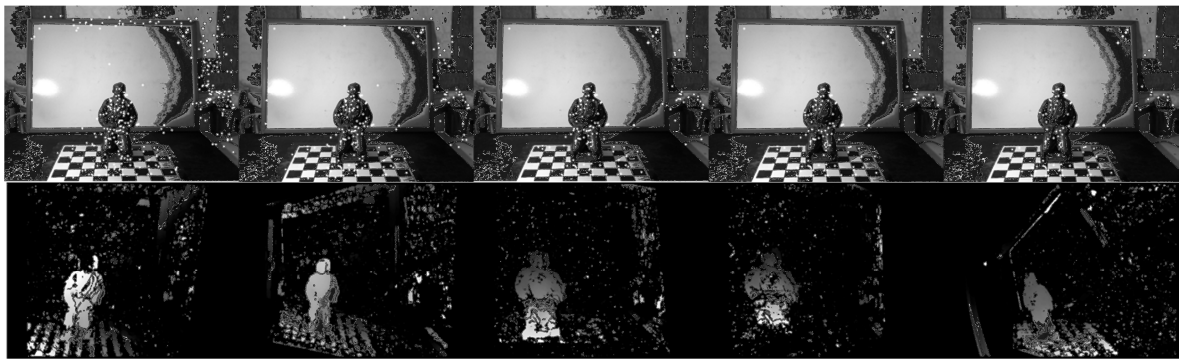


Figura 4.3.1: Mapas de disparidad y puntos de interés de la imagen para valores de la matriz Hessiana de izquierda a derecha de 100 ,300 ,500 ,700 y 1000.

Como se puede apreciar, el mapa de disparidad que más realismo aporta es el correspondiente a un valor de Hessiana de 500, y por ello se ha decidido utilizar este valor para continuar con el proyecto.

El resto de valores produce valores erróneos ya sea porque hay un exceso de puntos que se consideran de interés (el caso de un valor de Hessiana bajo) y no transforma correctamente el par estéreo, o bien porque hay un número bajo de puntos que deforma mucho el par estéreo.

Como en todos los módulos se obtienen los tiempos de cómputo en función del tamaño de imagen, para poder sacar finalmente conclusiones interesantes acerca del algoritmo.

Imagen	Tiempo (ms)	Parejas Encontradas	Puntos Válidos	Valor Hessiana
Set 1	372	302	139	500
Set 1	418	388	185	300
Set 1	344	260	124	700
Set 2	2936	734	335	700
Set 2	3188	928	425	500
Set 2	3543	1244	567	300
Set 5	691	1190	573	500
Set 5	836	1496	721	300
Set 5	1023	1990	951	100
Set 5	511	838	405	1000

Tabla 4.3.1: Tiempos de ejecución del programa en función de las imágenes calculadas.

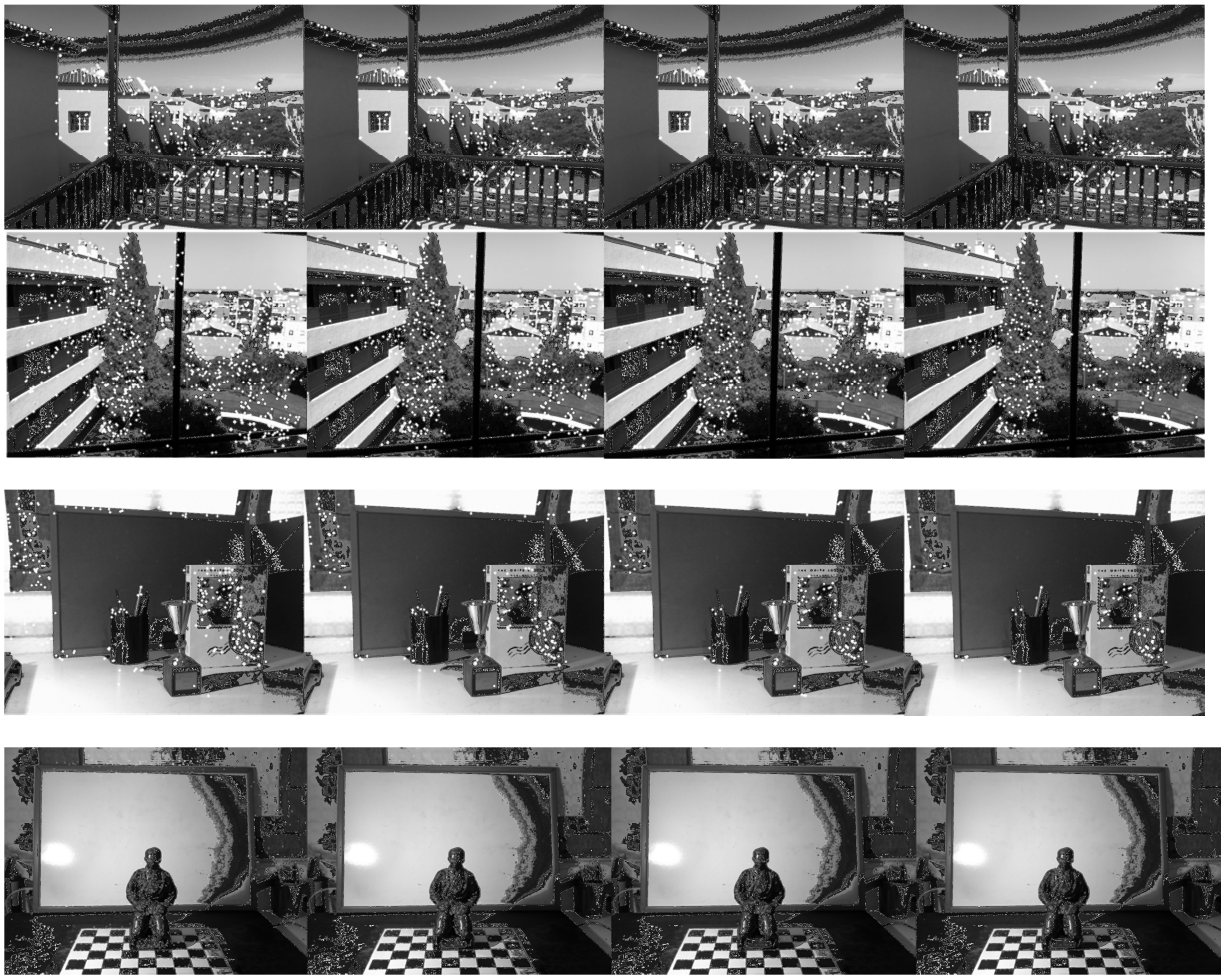


Figura 4.3.2: Diferentes Sets de imágenes con valores del parámetro variable de 100, 300, 500 y 700 (de izquierda a derecha) la última tira de imágenes se corresponde con una imagen de resolución 2592x1936. Como puede apreciarse tiene mayores puntos de interés que su semejante de la Figura 4.3.1 debido a la mayor resolución.

Como puede observarse en la tabla, a medida que se disminuye el valor que limita la aceptación o rechazo de los puntos característicos, los tiempos de cómputo aumentan en consonancia con el aumento de puntos que son aceptados. Y en el caso del Set 2 que se corresponde con una imagen de resolución elevada, los tiempos de cómputo, para igualdad de restricción con respecto al Set 1, se puede observar que crece al igual que el número de puntos encontrados, ya que se dispone de más información de la imagen.

Puede observarse como sucedía en el caso de la calibración, que para imágenes con resoluciones elevadas, los tiempos de cómputo se vuelven mucho más elevados, no tanto como en el caso de la calibración, pero ya que este es un proceso que tiene que repetirse para cada par de imágenes, el tiempo se vuelve un parámetro más crítico.

4.4 Resultados de la obtención de la Matriz Fundamental

En este breve apartado se presentan los diferentes métodos con los que se realiza la obtención de la matriz Fundamental.

Matriz Fundamental método RANSAC

$$F = \begin{bmatrix} 1.738 \cdot 10^{-7} & -4.012 \cdot 10^{-6} & 2.632 \cdot 10^{-3} \\ 5.774 \cdot 10^{-6} & 2.038 \cdot 10^{-7} & 1.327 \cdot 10^{-2} \\ -3.241 \cdot 10^{-3} & -1.532 \cdot 10^{-2} & 1 \end{bmatrix}$$

Matriz Fundamental método L MEDS

$$F = \begin{bmatrix} 6.043 \cdot 10^{-7} & -2.237 \cdot 10^{-5} & 8.005 \cdot 10^{-3} \\ 2.245 \cdot 10^{-5} & -1.776 \cdot 10^{-6} & -2.364 \cdot 10^{-2} \\ -8.689 \cdot 10^{-3} & 2.174 \cdot 10^{-2} & 1 \end{bmatrix}$$

Matriz Fundamental método 8 Puntos

$$F = \begin{bmatrix} -3.799 \cdot 10^{-8} & 2.794 \cdot 10^{-6} & -1.181 \cdot 10^{-3} \\ 1.147 \cdot 10^{-7} & 1.257 \cdot 10^{-6} & 2.147 \cdot 10^{-2} \\ 2.769 \cdot 10^{-4} & -2.447 \cdot 10^{-2} & 1 \end{bmatrix}$$

Matriz Fundamental método 7 Puntos

$$F = \begin{bmatrix} 6.541 \cdot 10^{-7} & -2.111 \cdot 10^{-5} & 8.354 \cdot 10^{-3} \\ 2.117 \cdot 10^{-5} & -1.954 \cdot 10^{-6} & -2.412 \cdot 10^{-2} \\ -8.987 \cdot 10^{-3} & 2.354 \cdot 10^{-2} & 1 \end{bmatrix}$$

De las matrices calculadas, las que mejores resultados nos aportan (mejores mapas de disparidad se obtienen) son el método RANSAC (como se predisponía a suponer ya que es el que se recomienda en [3]) y el método de 8 Puntos.

Los otros dos métodos, aunque son también válidos, nos dejan unos resultados de estéreo y por lo tanto de disparidad más pobres y de los que se pueden obtener peores resultados.

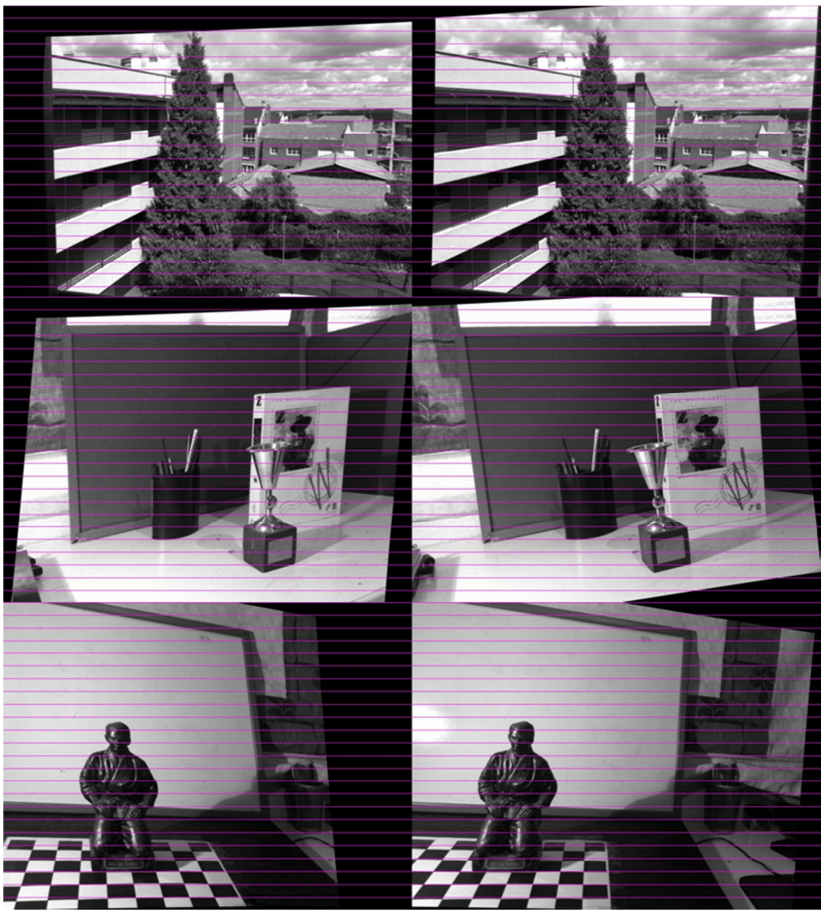


Figura 4.5.1: Ejemplos de rectificación estéreo de imágenes válidos.

4.5 Resultados de Rectificación Estéreo

En este apartado se pone de manifiesto otro factor clave a la hora de obtener la geometría tridimensional de la escena, ya que, es en este punto, donde se observa la diferencia real existente entre cada par estéreo, y desde el cual se puede obtener por primera vez un primer valor de la tercera variable del espacio tridimensional.

Se presentan un conjunto de imágenes en la Figura 4.5.1, en la que se puede comprobar como este paso es muy sensible a cómo se ha realizado la toma de las imágenes, por lo que se vuelve un factor importante que la toma de las imágenes se realice la forma más paralela posible para maximizar la zona común de visión

Figura 4.5.2: Ejemplo de mala Rectificación Estéreo

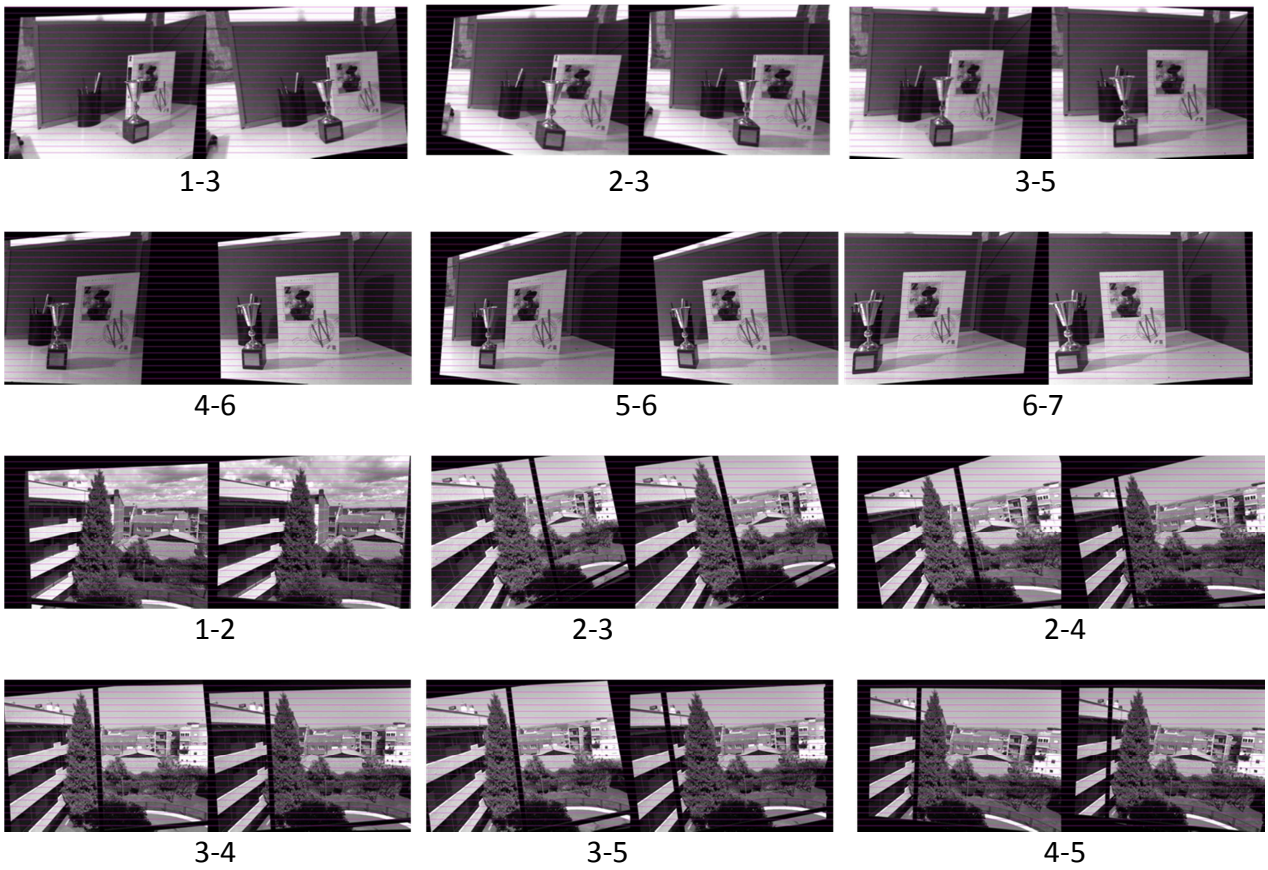


Figura 4.5.3: Pares estereoscópicos para distintos conjuntos de imágenes. Se puede observar como, por lo general, las imágenes consecutivas aportan mayores zonas comunes de rectificación estéreo que los pares salteados.

Para una correcta rectificación estéreo es necesario que exista una elevada cantidad de puntos en común, por lo que la fase de búsqueda de correspondencias es muy importante. Por consiguiente una variable de limitación para decidir si un par estéreo es correcto o no, para que el programa pueda hacerlo de forma automática, se limita la validez de las rectificaciones estéreo de los pares estereoscópicos a aquellas en las que exista un elevado valor de correspondencias encontradas, porque como se ve en las imágenes de la Figura 4.5.2, se puede observar cómo se deforma la imagen hasta extremos que no permiten obtener unos resultados aceptables.

Por lo general, se puede observar que este apartado es bastante sensible a alteraciones y a las posiciones de la cámara, por ello, por lo general la rectificación estéreo de imágenes no consecutivas no suele arrojar valores adecuados como se observa en la Figura 4.5.3.

Los tiempos de cómputo de este módulo se reflejan en la Tabla 4.5.1

Imágenes	Set 1	Set 2	Set 3	Set 4	Set 5	Set 6
Tiempo (ms)	135.4	419	142	412	151	147

Tabla 4.5.1: Tiempos de cómputo para el módulo de la rectificación estéreo y obtención de mapas de disparidad.

Como puede apreciarse, el tiempo de cálculo necesario para las imágenes de mayores resoluciones, es mucho mayor que el necesario para imágenes de resolución media.

En esta tabla se representan los tiempos de cómputo de la rectificación estéreo en conjunto con la del cálculo de mapas de disparidad, ya que forman parte de la misma función, y resulta más interesante y sencillo obtenerlo de esta manera.

4.6 Resultados de Mapas de Disparidad y Puntos 3D

Este es el último paso de obtención de parámetros necesarios para la reconstrucción. Con estos mapas tenemos ya prácticamente todo lo que necesitamos para obtener las coordenadas 3D de los puntos.

Aquí se presentan experimentos relativos a la variación de los parámetros que influyen a la hora de conseguir unos mapas de disparidad lo más fiables posibles.

Como vimos en el apartado anterior de metodología de trabajo, los parámetros más significantes que podían variarse en los mapas de disparidad, eran tres: mínima disparidad, número de disparidades y tamaño de ventana de cálculo de correlación.

En la Figura 4.6.1 se presentan resultados de consecuencia de la variación de la disparidad mínima, en la Figura 4.6.2 los relativos a la variación del número de disparidades y en la Figura 4.6.3 se presentan los relativos a la variación del tamaño de ventana.

Como se puede observar en la Figura 4.6.1, debido a que los pares de imágenes se han obtenido lo más paralelamente posible, a medida que nos alejamos del valor 0 de esta variable, los resultados se van volviendo muy pobres. En el caso en el que aumenta de forma positiva este valor se puede apreciar como se empieza a llenar la imagen de más valores oscuros (mayor valor de escala de grises). Por el contrario, cuando este valor se aleja de forma negativa, el efecto es el contrario, volviéndose la imagen más clara (valores de escala de grises menores) y aunque a simple vista de la sensación de tener mejores resultados y obtener mayor información, ésta se ve alterada ya que la escala se ve reducida en valores que puede tomar, perdiendo así mucha precisión.

En el caso de la Figura 4.6.2 se puede apreciar muy bien que a medida que aumenta el número de disparidades, los resultados pierden escalabilidad, esto es, que a medida que aumentamos este valor, los píxeles cercanos se consideran que tienen la misma profundidad que sus vecinos, por lo que en caso de buena rectificación estéreo,

tener un valor elevado de esta variable nos hace perder mucha precisión de profundidades. Por el contrario, si la rectificación se es pobre, se vuelve necesario aumentar el valor de esta variable para tener un resultado todo lo aceptable que permite la mala rectificación.

Por últimos, en la Figura 4.6.3 se aprecia como a medida que se aumenta la variable las imágenes tienen menos ruido, y esto es debido a, como se explicó en el apartado 3 de este proyecto, la búsqueda de correlaciones se realiza en ventanas de mayor tamaño, suavizando las mismas, por lo que al aumentar este valor, se suaviza mucho la imagen a costa de perder precisión.

Si la rectificación es muy buena (como se conseguiría con un par estéreo de cámaras fijo y paralelo) se podrían establecer valores bajos de esta variable, y así tener unos resultados francamente buenos de profundidad.

Como conclusión de este apartado, se ha decidido elegir un valor de disparidad mínima de 0, ya que supuestamente se toman las capturas de forma lo más paralela posible. Un valor de número de disparidades de 64, ya que es un valor intermedio que nos aporta una solución relativamente buena para los distintos ejemplos realizados y un tamaño de ventana de correlación de 15.

Lo ideal sería que para cada conjunto de imágenes se realizase un estudio de qué valores de variables son los óptimos y establecer los mismos para cada conjunto en particular.

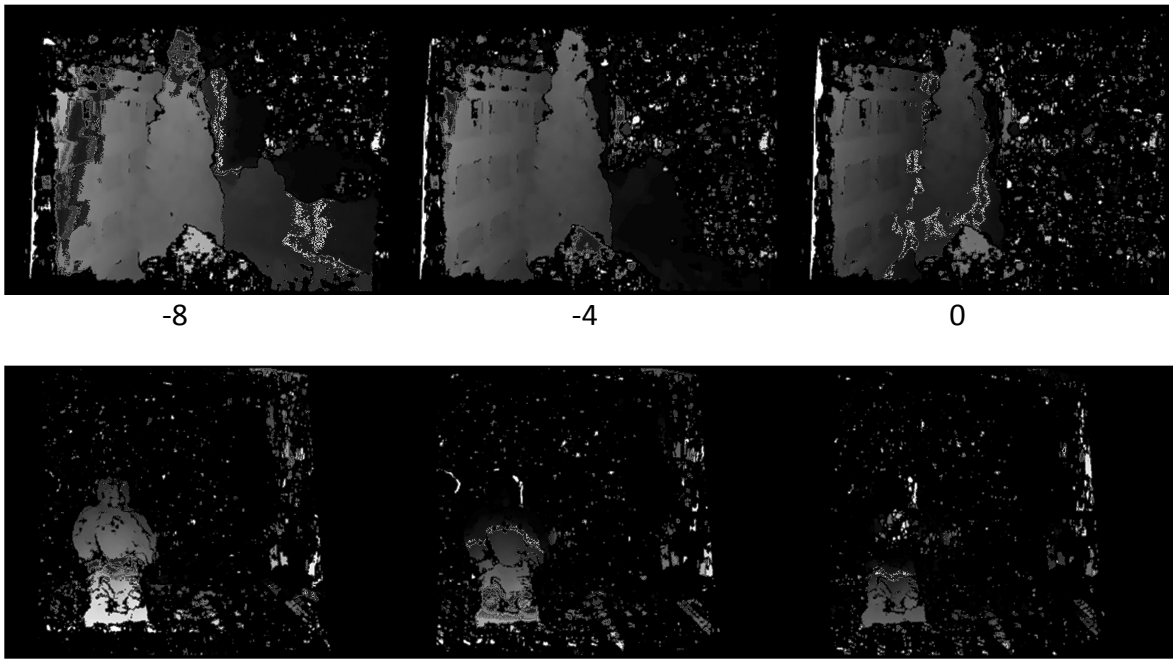


Figura 4.6.1: Ejemplos de variación de la disparidad mínima calculable, con los demás parámetros constantes.

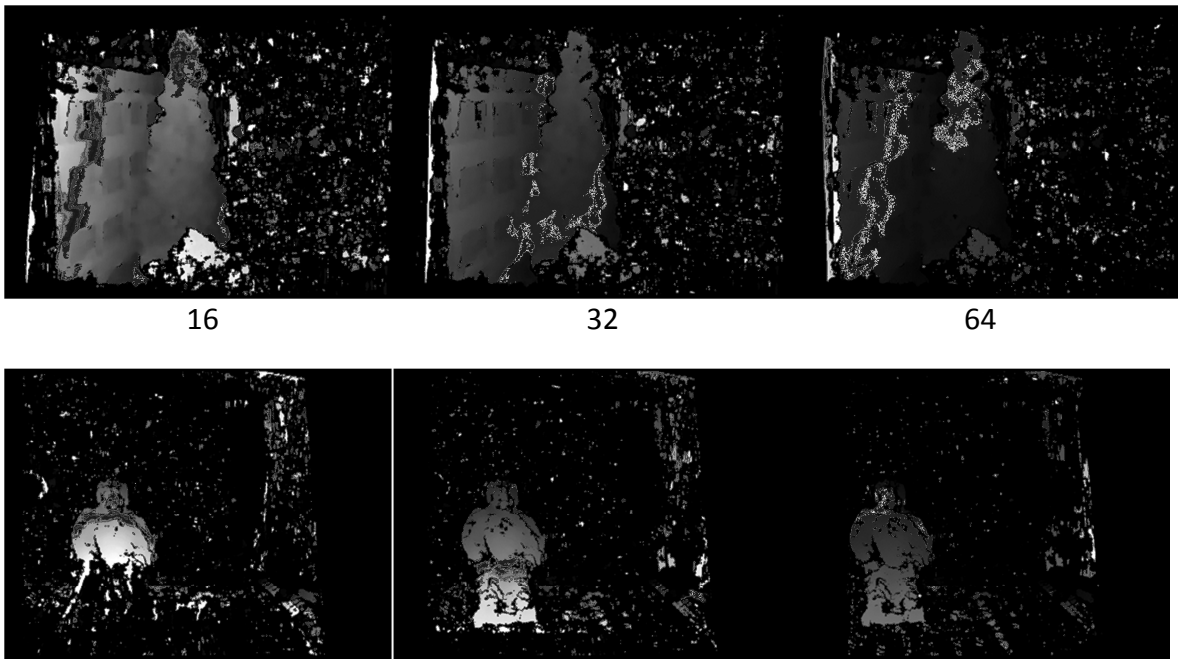
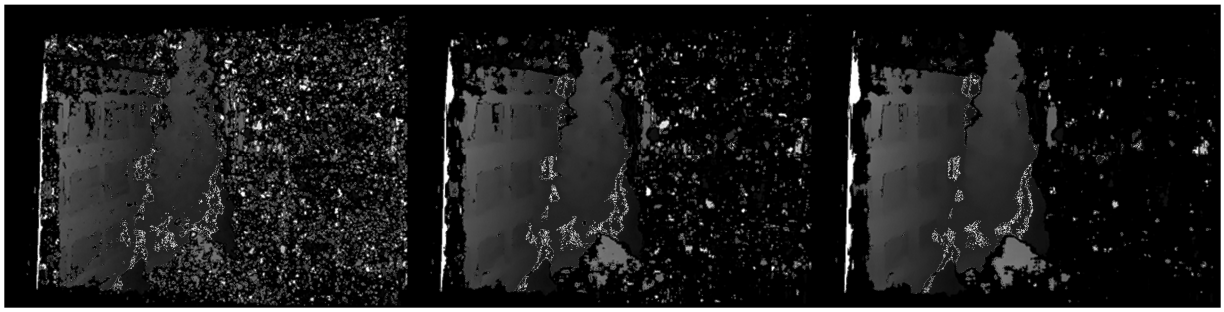


Figura 4.6.2: Representación de mapas de disparidad variando el número de disparidades



9 15 21

9 15 19 21

Figura 4.6.3: Mapas de disparidad obtenidos variando el tamaño de ventana de cálculo de correlación.

En la Figura 4.6.4 se muestran resultados obtenidos de los distintos conjuntos de imágenes realizados con los parámetros seleccionados prefijados. En algunos de ellos se puede observar que los resultados no son muy buenos, pero no solo se ve afectado por la elección de estos parámetros, sino que también por los pasos anteriores del algoritmo, así como por la toma misma de las imágenes con la cámara.

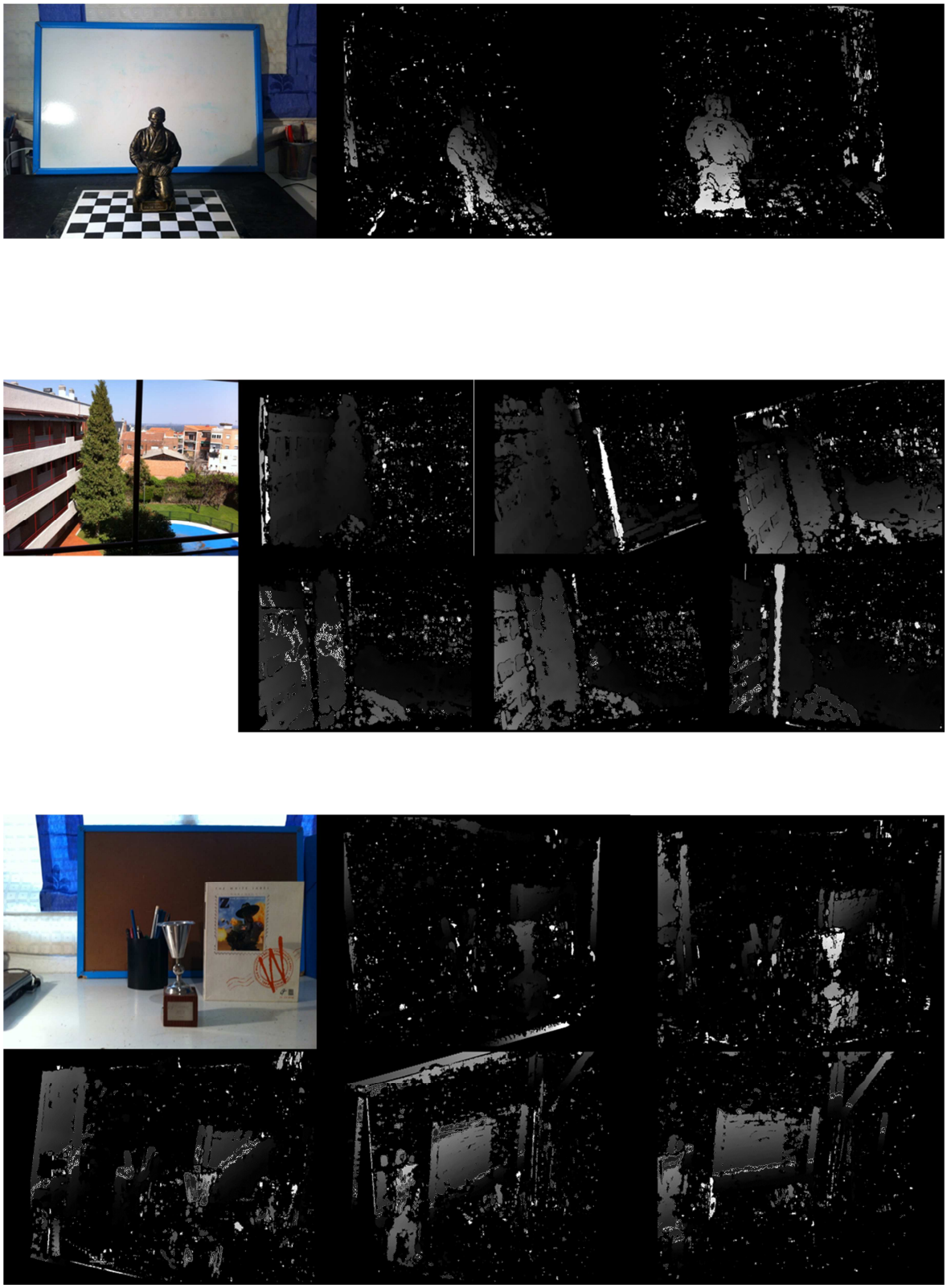


Figura 4.6.4: Mapas de disparidad para distintos pares estéreo de imágenes de diversos Sets.

4.7 Resultados de la Reconstrucción

Este es el último paso para la finalización del sistema. En él se toman como entrada las coordenadas 3D calculadas previamente y lo único que se necesita es representar la nube de puntos en un gráfico 3D.

En las siguientes figuras se muestran algunos de los resultados obtenidos mediante un programa para representar dichos puntos.

La calidad de la reconstrucción depende del número de imágenes utilizadas, ya que de cada par de imágenes analizado se obtienen un conjunto de coordenadas 3D, por lo que a mayor número de imágenes más resolución tendrá la reconstrucción.

Las zonas sin reconstruir se corresponden con oclusiones producidas de las imágenes. Para evitar estas oclusiones sería necesario realizar una elevada captura de imágenes con todos los ángulos para que ningún punto se encuentre oculto.

Otro de los motivos de que las imágenes obtenidas tengan zonas sin reconstruir, se debe a la calidad de los mapas de disparidad, en los que las zonas negras se consideran disparidades que tienen a una distancia infinita, por lo que los píxels correspondientes a esas zonas no se representan en la imagen.

Como se puede comprobar existen píxeles que se encuentran en zonas que no se corresponden con su lugar en la realidad, se debe a que en el mapa de profundidad, debido a que esos puntos no son característicos, su disparidad no es fiable.

Figura 4.7.1: Resultados de la reconstrucción de set_reconstrucción 1

Figura 4.7.2: Resultados de la reconstrucción de set_reconstrucción 2

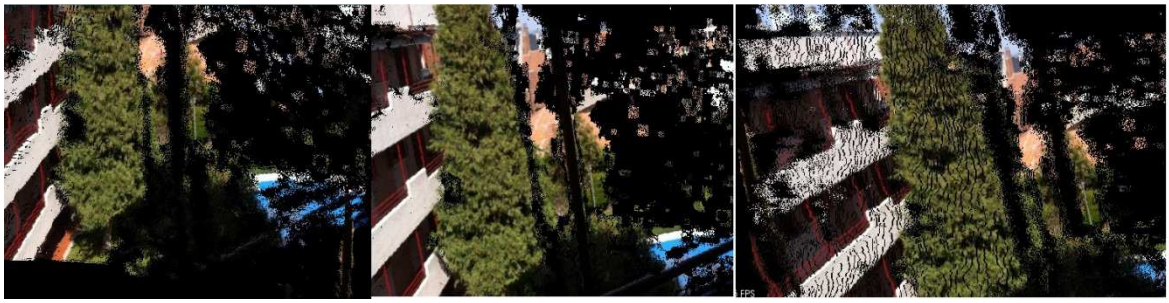


Figura 4.7.3: Resultados de reconstrucción de set 5

Figura 4.7.4: Resultados de reconstrucción de set 3

Como se ha comentado con anterioridad, los resultados tienen gran cantidad de errores y zonas sin representar. Pero aun así se puede percibir la tridimensionalidad existentente en las imágenes.

Se ha procedido a realizar una serie de capturas tras realizar movimientos en el visualizador 3D con el fin de percibir la profundidad existente.

Capítulo 5: Conclusiones y propuestas futuras

En este último apartado se pondrán de manifiesto las principales conclusiones que se han obtenido de la realización de este proyecto, así como una serie de ideas que podrían llegar a ser interesante estudiar para realizar mejoras en el modelo o incluso aplicaciones en las que podrían llegar a utilizarse estos conocimientos.

5.1 Conclusiones

Tras realizar un análisis del proceso de reconstrucción, así como de los resultados obtenidos en este proyecto, se sacan las siguientes conclusiones:

- Tras observar los tiempos relativos al desarrollo del proyecto, se puede comprobar que los tiempos necesarios para realizar el procesado de imágenes de gran resolución es excesivamente elevado en oposición con la calidad del resultado final obtenido, por lo que se puede determinar que una adquisición de imágenes de resolución media es suficientemente bueno para obtener reconstrucciones aceptables.

- Debido a que los tiempos de cómputo necesarios para la ejecución del módulo de reconstrucción son elevados y han de realizarse para cada par de imágenes, se puede concluir que este sistema no tiene validez para aplicaciones en tiempo real.

- Acerca de la calidad de la reconstrucción, se puede comprobar que la robustez del sistema ante una adquisición de imágenes de forma aleatoria y no realizada de forma cuidadosa, es muy pobre, llegando incluso a no obtener un resultado aceptable, por lo que se demuestra la fragilidad de este sistema.

- Los procesos intermedios de búsqueda de correspondencias y mapas de disparidad (ambos extremadamente importantes para obtener un buen resultado final) cobran gran importancia, y es indispensable que se obtengan resultados lo más óptimos posibles, nuevamente, este hecho, nos indica la fragilidad del sistema completo.

- Por último cabe destacar que el proceso y los resultados finales serían mucho mejores si se hubiese realizado la adquisición de las imágenes con un par estéreo fijo, ya que los parámetros como la matriz fundamental y la matriz de reproyección serían siempre los mismos, facilitando así muchos cálculos dentro del algoritmo, modificándose incluso algunas de las funciones, consiguiendo también optimizar los tiempos de procesado y los resultados de correspondencias y mapas de disparidad.

5.2 Propuestas Futuras

En este apartado se ponen de manifiesto algunas ideas que han ido surgiendo a lo largo del desarrollo del proyecto para mejorar el mismo, así como otras líneas de investigación relacionadas que podrían dar lugar a nuevos proyectos en este ámbito.

- Implementación de una solución en la que se realice una autocalibración de la escena mediante información contenida en las propias imágenes capturadas.

- Implementación de un sistema de visión estéreo embebido en un robot, dotando al mismo de autosuficiencia para el movimiento autónomo.

- Estudio de nuevos algoritmos y fórmulas que permitan la reconstrucción de la escena en tiempo real, aportando así una gran funcionalidad de este tipo de solución.

- Optimización y mejora de la solución propuesta en este proyecto, incluyendo una etapa de postprocesado en la que se cree una malla mediante triangulación que permita la aplicación posterior de una máscara sobre dicha malla y así obtener una sensación más limpia de reconstrucción.

Capítulo 6: Bibliografía

[1] Richard Szelinski " *Computer Visión Algorithms and applications*" Springer Verlang, 2011

[2] Robert Laganière " *OpenCV 2 Computer Vision Application Programming Cookbook*" Packt Publishing, 2011

[3] Gary Bradski & Adrian Kaehler "*Learning OpenCV: OpenCV with the OpenCV library*" O´Really, 2008

[4] Forsyth, Ponce "*Computer Vision A Modern Approach"* Prentice Hall,2003

[5] Arturo de la Escalera "*Visión por computador*" Prentice Hall, 2006

[6] Zisserman A. Hartley R. "*Multiple view geometry in computer visión*" Cambridge, 2003

[7] Diego Aracena, Pedro Campos y Clésio Luis Tozzi "*Comparación de técnicas de calibración de cámaras digitales*" ,2005

[8] Tomás Rodríguez, Peter Sturm, Pau Gallardo "*Photorealistic 3D reconstruction from handheld cameras"* Machine vision and applications 2005

[9] Marteen Vergauwen and Luc Van Gool "*3D Acquisition*"

[10] Zhengyou Zhang "*A flexible new technique for camera calibration*", 1998

[11] Zhengyou Zhang "*Flexible camera calibration by viewing a plane from unknown orientations*", 1999

[12] Tsai R. "*A versatile camera calibration technique for high-accuracy 3D machine visión metrology using off-the-self TV cameras and lenses*", 1987

[13] Faugueras O. "*Three Dimensional computer visión- A Geometric viewpoint*", 1993

[14]Marc Pollefeys, Reinhard Koch, Luc Van Gool "*Self-Calibrating and metric reconstruction in spite of varying and unknown intrinsic camera parameters*", 1998

[15] Christopher Evans "Notes on the OpenSURF Library"

[16] Mikolajczyk K. "*Detection of local features invariant to affine transformations*"

[17] Lowe D. "*Distinctive image features from scale-invariant keypoints*" International journal of computer vision, 2004

[18] Reinhard Koch,Marc Pollefeys , Luc Van Gool "*Realistic 3D scene modeling from uncalibrated image sequences*"

[19] Theo Moons, Luc Van Gool, Martin Vergauwen " *3D Reconstrution from multiple images*"

[20]Herbert Bay, Andreas Ess, Tinne Tuytelaars, Luc Van Gool "*Speed-Up Robust Features (SURF)*" Proceeding of the ninth European conference of computer vision, 2006

[21] Richard I. Hartley and Peter Sturm "*Triangulation*"

[22] W. Triggs, R. Hartley "*Vision Algorithms: Theory and practice*" Springer Verlag, 2000

[23] W. H. Press, S.A. Kolsky, W.T. Vetterling " *Numerical récipes in C: The art of scientific computing*" Cambridge University Press, 1992

[24] "Sitio web de OpenCv" www.willowgarage.com

ANEXO A: Diagramas de Flujo

Diagrama del módulo de calibración

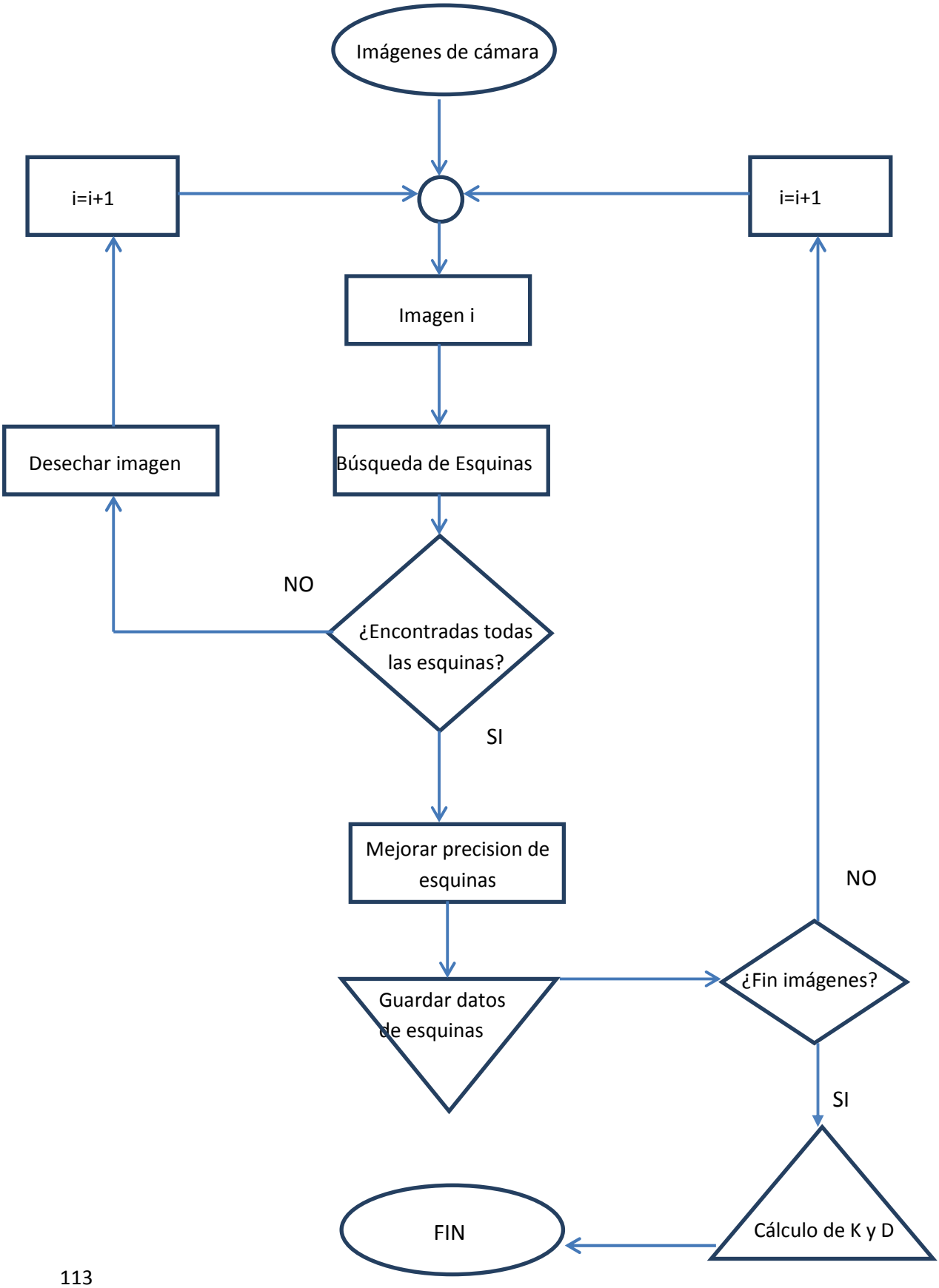


Diagrama del módulo de reconstrucción